Checking Framework Interactions with Relationships

Ciera Jaspan **Jonathan Aldrich**

December 2008
CMU-ISR-08-140

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Software frameworks impose constraints on how plugins may interact with them. Many of hese constraints involve multiple objects, are temporal, and depend on runtime values. Additionally, they are difficult to specify because they are non-local and may break behavioral subtyping. This work presents *relationships* as a means for specifying framework constraints, and it presents a formal description and implementation of a static analysis to find constraint violations in plugin code. We define three variants of this analysis: one is sound, one is complete, and one provides compromise of the two. We prove soundness and completeness for the appropriate variants, and we show how the compromise variant works on examples from real-world programs. This allows the user to select the option which is the most cost-effective in practice with regard to the number of false positives and false negatives.

This work was supported in part by a fellowship from Los Alamos National Laboratory, DARPA contract HR00110710019, and Army Research Office grant number DAAD19-02-1-0389 entitled Perpetually Available and Secure Information Systems.

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE
DEC 2008

2. REPORT TYPE

3. DATES COVERED
00-00-2008 to 00-00-2008

4. TITLE AND SUBTITLE
Checking Framework Interactions with Relationships

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:

a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	17. LIMITATION OF ABSTRACT **Same as Report (SAR)**	18. NUMBER OF PAGES **110**	19a. NAME OF RESPONSIBLE PERSON

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Keywords: software frameworks, relationships, static analysis, verification

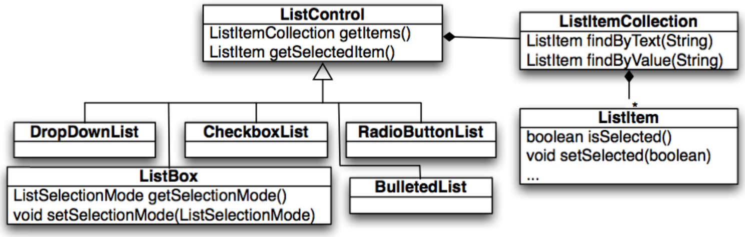

ListControl
ListItemCollection getItems()
ListItem getSelectedItem()
ListItemCollection
ListItem findByText(String)
ListItem findByValue(String)
*
ListItem
boolean isSelected()
void setSelected(boolean)
...
DropDownList
CheckboxList
RadioButtonList
ListBox
ListSelectionMode getSelectionMode()
void setSelectionMode(ListSelectionMode)
BulletedList

Figure 1: ASP.NET ListControl Class Diagram

1 Introduction

Object-oriented frameworks have brought many benefits to software development, including reusable codebases, extensible systems, and encapsulation of quality attributes. However, frameworks are used at a high cost; they are complex and difficult to learn [11]. This is partially due to the complexity of the semantic constraints they place on the *plugins* that utilize them.

As an example, consider a constraint in the ASP.NET web application framework. The ASP.NET framework allows developers to create web pages with user interface controls on them. These controls can be manipulated programatically through callbacks provided by the framework. A developer can write code that responds to control events, adds and removes controls, and changes the state of controls.

One task that a developer might want to perform is to programmatically change the selection of a drop down list. The ASP.NET framework provides the relevant pieces, as shown in Figure 1[1]. Notice that if the developer wants to change the selection of a `DropDownList` (or any other derived `ListControl`), she has to access the individual `ListItems` through the `ListItemCollection` and change the selection using `setSelected`. Based on this information, she might naïvely change the selection as shown in Listing 1. Her expectation is that the framework will see that she has selected a new item and will change the selection accordingly.

When the developer runs this code, she will get the error shown in Figure 2. The error message clearly describes the problem; a `DropDownList` had more than one item selected. This error is due to the fact that the developer did not de-select the previously selected item, and, by design, the framework does not do this automatically. While an experienced developer will realize that this was the problem, an inexperienced developer might be confused because she did not select multiple items.

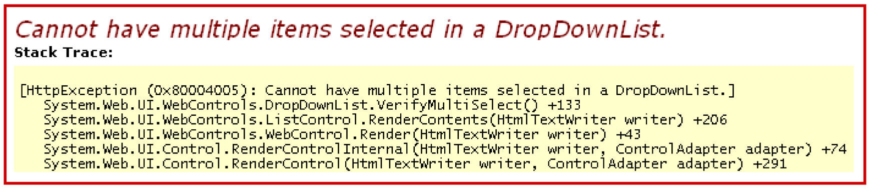
The stack trace in Figure 2 is even more interesting because it does not point to the code where the developer made the selection. In fact, the entire stack trace is from framework code; there is no plugin code referenced at all! At runtime, the framework called the plugin developer's code in Listing 1, this code ran and returned to the framework, and then the framework discovered the error. To make matters worse, the program control could go back and forth several times before finally reaching the check that triggered the exception. Since the developer doesn't know exactly where the problem occurred, or even what object it occurred on, she must search her code by hand to find the erroneous selection.

[1] To make this code more accessible to those unfamiliar with C#, we are using traditional getter/setter syntax rather than properties.

Listing 1: Incorrect selection for a DropDownList

```
DropDownList list;

private void Page_Load(object sender, EventArgs e)
{
   ListItem newSel;
   newSel = list.getItems().findByValue("foo");
   newSel.setSelected(true);
}
```



```
Cannot have multiple items selected in a DropDownList.
Stack Trace:

[HttpException (0x80004005): Cannot have multiple items selected in a DropDownList.]
   System.Web.UI.WebControls.DropDownList.VerifyMultiSelect() +133
   System.Web.UI.WebControls.ListControl.RenderContents(HtmlTextWriter writer) +206
   System.Web.UI.WebControls.WebControl.Render(HtmlTextWriter writer) +43
   System.Web.UI.Control.RenderControlInternal(HtmlTextWriter writer, ControlAdapter adapter) +74
   System.Web.UI.Control.RenderControl(HtmlTextWriter writer, ControlAdapter adapter) +291
```

Figure 2: Error with partial stack trace from ASP.NET

The correct code for this task is in Listing 2. In this code snippet, the developer de-selects the currently selected item before selecting a new item.

This example, and many others we have found on the ASP.NET developer forum, shows three interesting properties of framework constraints.

Framework constraints involve multiple classes and objects. Listing 1 references three objects, and Listing 2 required four objects to make the proper selection. The framework code that the plugin used was located in four classes.

Framework constraints are non-local. While the `DropDownList` was the class that checked the constraint (as seen by the stack trace), the constraint itself was on the methods of `ListItem`. However, the `ListItem` class is not aware of the `DropDownList` class or even that it is within a `ListControl` at all, and therefore it should not be responsible for enforcing the constraint. The non-local nature of these constraints also makes them difficult to document, as it is unclear where the documentation should go so that the plugin developer will discover it. In this example, had the framework developer placed the relevant documentation in the `DropDownList`, the plugin devel-

Listing 2: Correctly selecting an item using the ASP.NET API

```
DropDownList list;

private void Page_Load(object sender, EventArgs e)
{
   ListItem newSel, oldSel;
   oldSel = list.getSelectedItem();
   oldSel.setSelected(false);
   newSel = list.getItems().findByValue("foo");
   newSel.setSelected(true);
}
```

Listing 3: Selecting on the wrong DropDownList

```
DropDownList listA;
DropDownList listB;

private void Page_Load(object sender, EventArgs e)
{
  ListItem newSel, oldSel;
  oldSel = listA.getSelectedItem();
  oldSel.setSelected(false);
  newSel = listB.getItems().findByValue("foo");
  newSel.setSelected(true);
}
```

oper might still not find it because she was using methods of the `ListItem` class.

Framework constraints have semantic properties. Framework constraints are not only about structural concerns such as method naming conventions or types; the developer must also be aware of semantic properties of the constraint. There are several semantic properties shown by this example. First, the plugin developer had to be aware of which objects she was using to avoid the problem in Listing 3. In this example, the developer called the correct operations, but on the wrong objects. She also had to be aware of the primitive values (such as `true` or `false`) she used on the calls to change the selection. Finally, she had to be aware of the ordering of the operations. In Listing 2, had she swapped lines 6 and 7 with lines 8 and 9, she would have caused unexpected runtime behavior where the selection change does not occur. This behavior occurs because `getSelectedItem` returns the first selected `ListItem` that it finds in the `DropDownList`, and that may be the newly selected item rather than the old item.

In previous work [10], we proposed a preliminary specification approach and sketched a hypothetical analysis to discover mismatches between the plugin code and the declared constraints of the framework. The previous work primarily discussed the requirements for such a system and explored a prototype specification. In this paper, we make three contributions:

1. We show that the concept of developer-defined relations across objects captures the primary programming model used to interact with frameworks. We use these relations to specify framework constraints in a concise manner. (Section 2)
2. We propose (Section 3) and formally define (Section 4) a static analysis that detects where a plugin violates framework constraints. We define three variants of this analysis: a sound variant, a complete variant, and a third variant that is neither sound nor complete. We prove soundness and completeness for the appropriate variants, and we argue that the third variant is a better compromise for practical use. Additionally, there are only minor differences between the variants, so it is it simple to swap between them.
3. We implemented the compromise variant of the analysis within the Eclipse IDE and ran it on code based on examples from framework help forums. We show that the constraints capture the properties described and that the compromise variant can handle real-world code with relatively few false positives and false negatives. (Section 5)

2 Developer-defined Relations over Objects

WWhen a developer programs to a framework, the primary task is not about creating new objects or data. In many cases, programming in this environment is about *manipulating the abstract associations between existing objects*. Every time the plugin receives a callback from the framework, it is implicitly notified of the current associations between objects. As the plugin calls framework methods, the framework changes these associations, and the plugin learns more about how the objects relate. Every method call, field access, or test gives the plugin more information. Even when the plugin needs to create a new object, it is frequently done by calling abstract factory methods that set up the object and its relationships with other objects.

The ASP.NET framework exemplifies this means of interaction. In the `DropDownList` example, all the objects are provided by the framework, and the plugin simply changes their relationships with each other through calls to the framework. In fact, the `DropDownList` itself, and the data within it, is frequently set up using dependency injection, a mechanism in which the framework populates the fields of the plugin based on an external configuration file [7]. This may be done in several stages, with the framework notifying the plugin as it completes each stage using a callback. When using dependency injection, the plugin simply receives and manipulates preconfigured objects.

Since the primary mechanism of interaction is based on manipulating relationships between objects, we will model it formally using a mathematical relation. A *relation* is a named, mathematical relation on several types τ.

$$\mathsf{Relation} ::= \mathsf{name} \rightarrow \tau_1 \times \ldots \times \tau_n$$

A *relationship* is a single tuple in a relation, represented as

$$\mathsf{Relationship} ::= \mathsf{name}(\ell_1, \ldots, \ell_n)$$

where ℓ is a static representation of a runtime object.

In this section, we introduce three specification constructs based on relationships. The first construct, *relationship effects*, specify how framework operations change associations between objects. The second construct, *constraints*, uses relationships to specify the non-local constraints on framework operations. Finally, *relation inference rules* specify how relationships can be inferred based on the current state of other relationships, regardless of what operations are used.

2.1 Relationship Effects

Relationship effects specify changes to the relations that occur after calling a framework method. The framework developer annotates the framework methods with information about how the calling object, parameters, and return value are related (or not related) after a call to the method. These annotations describe additions and removals of relationships from a relation. For example, the annotation @Item({item, list}, ADD) creates an "Item" relationship between `item` and `list`, while @Item({item, list}, REMOVE) removes this relationship[2]. Relationship effects may refer to the

[2] We are presenting a simplified version of the syntax for readability purposes. The correct Java syntax for the add annotation appears as @Item(params={"item", "list"}, effect=ADD). This is the syntax used in the implementation.

Listing 4: Relations for the `ListControl` API. Every relation must define the properties `params`, `effect`, and `test`

```
@Relation({ListItem.class, ListControl.class})
public @interface Child {
    public String[] params;
    public Effect effect;
    public String test = "";
}
```

parameters, the receiver object, and the return value of a method. They may also refer to primitive values. Additionally, parameters can be wild-carded, so `@Item({*, list}, REMOVE)` removes *all* the "Item" relationships between `list` and any other object.

In addition to the `ADD` and `REMOVE` effects, a `TEST` effect uses a parameter to determine whether to add or remove a relationship. For example, we might annotate the method `List.contains(Object obj)` with `@Item({obj, this}, TEST, return)` to signify that this relationship is added when the value of `return` is true and removed when the value of `return` is false.

As relations are user-defined, they have no predefined semantics. Any hierarchy or ownership present, such as "Child" or "Item" relations, is only inserted by the framework developer. In fact, relationships do not have to reflect *any* reference paths found in the heap, but may exist only as an abstraction to the developer. This allows relations to be treated as an abstraction independent from code, and even allows the same relation to be used across frameworks.

To define a new relation, the framework developer creates an annotation and uses the meta-annotation `@Relation` to signify it as a relation over specific types. Listing 4 shows a sample definition of the `Child` relation from the `DropDownList` example.

Once the framework developer defines the desired relations, they can be used as relationship effects, as shown in Listing 5. These annotations allow tools to track relationship effects through the plugin code at compile time. Listing 6 shows a snippet from a plugin, along with the current relationships after each instruction. For example, after line 4 in Listing 6, we learn the relationships in displayed in line 5 based on the effects declared for in Listing 5, lines 7-9. This information, the *relationship context*, provides us with an abstract, semantic context that each instruction resides in. In the next section, we use this context to check the semantic parts of framework constraints.

2.2 Constraints

Constraints use relationships in logical predicates to specify non-local preconditions of framework operations. They are written as class-level annotations, but as constraints are non-local, they can constrain the operations on any other class. Three examples of constraints on the `DropDownList` class are in Listing 7. As the examples show, a constraint has four parts:

1. *operation*: This is a signature of an operation to be constrained, such as a method call, constructor call, or even a tag signaling the end of a method. Notice that these operations may constrain operations on another class.
2. *trigger predicate*: This is a logical predicate over relationships. The plugin's relationship context must show this predicate to be true for this constraint to be triggered. If not, the

Listing 5: Partial `ListControl` API with Relation annotations

```
public class ListControl {
    @List({return, this}, ADD)
    public ListItemCollection getItems();

    //After this call, we know two pieces of information.
    //The returned item is selected, and it is a child of this
    @Child({return, this}, ADD)
    @Selected({return}, ADD)
    public ListItem getSelectedItem();
}
public class ListItem {
    //if the return is true, then we know we have a selected item
    //if it is false, we know it was not selected.
    @Selected({this}, TEST, return)
    public boolean isSelected();

    @Selected({this}, TEST, select)
    public void setSelected(boolean select);

    @Text({return, this}, ADD)
    public String getText();

    //When we call setText, remove any previous Text relationships,
    //then add one for text
    @Text({*, this}, REMOVE)
    @Text({text, this}, ADD)
    public void setText(String text);
}
public class ListItemCollection
    @Item({item, this}, REMOVE)
    public void remove(ListItem item);

    @Item({item, this}, ADD)
    public void add(ListItem item);

    @Item({item, this}, TEST, return)
    public boolean contains(ListItem item);

    @Item({item, this}, ADD)
    @Text({text, return}, ADD)
    public ListItem findByText(String text);

    //if we had any items before this, remove them after this call
    @Item({*, this}, REMOVE)
    public void clear();
}
```

Listing 6: Comments showing how the relationship context changes after each instruction

```
DropDownList ddl = ...;
ListItemCollection coll;
ListItem newSel, oldSel;
oldSel = ddl.getSelectedItem();
    //Child(oldSel, ddl), Selected(oldSel)
oldSel.setSelected(false);
    //Child(oldSel, ddl), !Selected(oldSel)
coll = ddl.getItems();
    //Child(oldSel, ddl), !Selected(oldSel), List(coll, ddl)
newSel = coll.findByText("foo");
    //Child(oldSel, ddl), !Selected(oldSel), List(coll, ddl),
    //Item(newSel, coll), Text("foo", newSel)
```

Listing 7: DropDownList Selection Constraints and Inferred Relationships

```
@Constraint(
    op="ListItem.setSelected(boolean select)",
    trigger="select == false and Child(this, ctrl) and
        ctrl instanceof DropDownList",
    requires="Selected(this)",
    effect={"!CorrectlySelected(ctrl)"}
)

@Constraint(
    op="ListItem.setSelected(boolean select)",
    trigger="select == true and Child(this, ctrl) and
        ctrl instanceof DropDownList",
    requires="!CorrectlySelected(ctrl)",
    effect={"CorrectlySelected(ctrl)"}
)

@Constraint(
    op="end-of-method",
    trigger="ctrl instanceof DropDownList",
    requires="CorrectlySelected(ctrl)",
    effect={}
)
@Infer(
    trigger="List(list, ctrl) and Item(item, list)",
    infer={"Child(item, ctrl)"}
)
public class DropDownList {...}
```

constraint is ignored. While *operation* provides a syntactic trigger for the constraint, *trigger* provides the semantic trigger.

3. *requires predicate*: This is another logical predicate over relationships. If the constraint

is triggered, then this predicate must be true under the current relationship context. If the requires predicate is not true, this is a broken constraint and the analysis should signal an error in the plugin.

4. *effect list*: This is a list of relationship effects. These effects will only be applied if the constraint is triggered.

In the first example at the top of Listing 7, the constraint is checking that at every call to `ListItem.setSelected(boolean)`, if the relationship context shows that the argument is false, the receiver is a `Child` of a `ListControl`, and if the `ListControl` is a `DropDownList`, then it must also indicate that the `ListItem` is `Selected`. Additionally, the context will change so that the `DropDownList` is not `CorrectlySelected`. The second constraint is similar to the first and in enforces proper selection of `ListItems` in a `DropDownList`. The third constraint ensures that the method does not end in an improper state by utilizing the "end-of-method" instruction to trigger when a plugin callback is about to end.

In some cases, the relationships between objects are implicit. Consider the `ListItemCollection` from the `DropDownList` example. In this example, the framework developer would like to state that items in this list are in a `Child` relation with the `ListControl` parent. However, it does not make sense to annotate the `ListItemCollection` class with this information since `ListItemCollections` should not know about `ListControls`.

2.3 Inferred relationships

In some cases, the relationships between objects are implicit. Consider the `ListItemCollection` from the `DropDownList` example. In this example, the framework developer would like to state that items in this list are in a `Child` relation with the `ListControl` parent. However, it does not make sense to annotate the `ListItemCollection` class with this information since `ListItemCollections` should not know about `ListControls`.

Inferred relationships describe these implicit relationships that can be assumed at any time. In Listing 7, lines 23-26 show an example for inferring a `Child` relationship based on the relations `ListItemCollections` and `ListControls`. Whenever the relationship context can show that the "trigger" predicate is true, it can infer the relationship effects in the "infer" list. It is possible to produce inferred relationships that directly conflict with the relationship context. To prevent this, the semantics of inferred relationships is that they are ignored in the case of a conflict, that is, relationships from declared relationship effects and constraints have a higher precedence.

3 The Relation Analysis

We have designed and implemented a static analysis to track relationships through plugin code and check plugin code against framework constraints. The relation analysis is a branch-sensitive, forward dataflow analysis[3]. It is designed to work on a three address code representation of Java-like source. We assume that the analysis runs in a framework that provides all of these features. In this section, we will present the analysis data structures, the intuition behind the three variations of the analysis, and a discussion of their tradeoffs. Section 4 defines how the analysis runs on each instruction.

The relation analysis is dependent on several other analyses, including a boolean constant propagation analysis and an alias analysis. The relation analysis uses the constant propagation analysis for the TEST effect. For this purpose, the relation analysis assumes there is a function $\mathcal{B}$ to which it can pass a variable and learn whether the represented value is true, false, or unknown.

The relation analysis can use any alias analysis which implements a simple interface. First, it assumes there is a context $\mathcal{L}$ that given any variable $\mathtt{x}$, provides a finite set $\overline{\ell}$ of abstract locations that the variable might point to. Second, it assumes a context Γ_ℓ which maps every location ℓ to a type τ. The combination of these two contexts, $<\Gamma_\ell, \mathcal{L}>$ is represented as the alias lattice $\mathcal{A}$.

The alias lattice must be conservative in its abstraction of the heap, as defined by Definition 1.

Definition 1 (Abstraction of Alias Lattice)**.** *Assume that a heap* h *is defined as a set of source variables* $\mathtt{x}$ *which point to a runtime location* ℓ *of type* τ*. Let* H *be all the possible heaps at a particular program counter. An alias lattice* $<\Gamma_\ell, \mathcal{L}>$ *abstracts* H *at a program counter if and only if*

$$\begin{array}{l}
\forall\, h \in H \,.\, \mathrm{dom}(h) = \mathrm{dom}(\mathcal{L})\text{and} \\
\quad \forall\, (\mathtt{x}_1 \hookrightarrow \ell_1 : \tau_1) \in h \,.\, \forall\, (\mathtt{x}_2 \hookrightarrow \ell_2 : \tau_2) \in h \,. \\
\qquad \text{if } \mathtt{x}_1 \neq \mathtt{x}_2 \text{ and } \ell_1 = \ell_2 \text{ then} \\
\qquad\quad \ell' \in \mathcal{L}(\mathtt{x}_1) \text{ and } \ell' \in \mathcal{L}(\mathtt{x}_2) \text{ and } \tau_1 <: \Gamma_\ell(\ell') \\
\qquad \text{and} \\
\qquad \text{if } \mathtt{x}_1 \neq \mathtt{x}_2 \text{ and } \ell_1 \neq \ell_2 \text{ then} \\
\qquad\quad \ell_1' \in \mathcal{L}(\mathtt{x}_1) \text{ and } \ell_2' \in \mathcal{L}(\mathtt{x}_2) \text{ and } \ell_1' \neq \ell_2' \text{ and } \tau_1 <: \Gamma_\ell(\ell_1') \text{ and } \tau_2 <: \Gamma_\ell(\ell_2')
\end{array}$$

This definition ensures that if two variables alias under any heap, then the alias lattice will reflect that by putting the same location ℓ' into each of their location lists. Likewise, if any heap can determine that the two variables are not aliased, then the alias lattice will reflect this possibility as well by having a distinct location in each location set. The definition also ensures that the typing context Γ_ℓ has the most general type for a location.

As long as the alias analysis maintains the abstraction property and can provide the required interface, the relation analysis can be proven to be either sound or complete. Of course, a more precise alias analysis will increase the precision of the relation analysis.

[3] By branch-sensitive, we mean that the true and false branches of a conditional may receive different lattice information depending upon the condition. The transfer function on the condition is called twice, once assuming that the result is false, and once assuming that it is true. This is not a path-sensitive analysis; the branch condition is not saved for use after the branches merge together.

3.1 The Relationship State Lattice

We track the status of a relationship using the four-point dataflow lattice represented in Figure 3, where `unknown` represents either true or false and `bottom` is a special case used only inside the flow function. The relation analysis uses a tuple lattice which maps all relationships we want to track to a relationship state lattice element. We will represent this tuple lattice as ρ. We will say that ρ is *consistent* with an alias lattice $\mathcal{A}$ when the domain of ρ is equal to the set of relationships that are possible under $\mathcal{A}$.

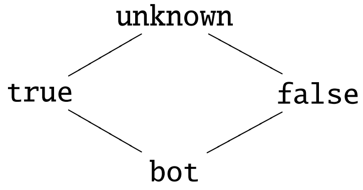


Figure 3: The simple lattice for a relationship

Notice that as more references enter the context, there are more possible relationships, and the height of ρ grows. Even so, the height is always finite as there is a finite number of locations and a finite number of relations. As the flow function is monotonic, the analysis always reaches a fix-point.

3.2 Flow Function

The analysis flow function is responsible for two tasks; it must check that a given operation is valid, and it must apply any specified relationship changes to the lattice. The flow function is defined as

$$f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathit{instr}) = \rho'$$

where $\mathcal{C}$ are all the constraints, $\mathcal{A}$ is the alias lattice, $\mathcal{B}$ is the boolean constant lattice, ρ is the starting relation lattice, ρ' is the ending relation lattice, and instr is the three-address code instruction on which we are running the analysis. The analysis goes through each constraint in $\mathcal{C}$ and checks for a match. It first checks to see whether the operation defined by the constraint matches the instruction, thus representing a syntactic match. It also checks to see whether ρ determines that the trigger of the constraint applies. If so, it has both a syntactic and semantic match, and it binds the specification variables to the locations that triggered the match.

Once the analysis has a match, two things must occur. First, it uses the bindings generated above to show that the required predicate of the constraint is true under ρ. If it is not true, then the analysis reports an error on instr. Second, the analysis must use the same bindings to produce ρ' by applying the relationship effects.

3.3 Soundness and Completeness

Soundness and completeness allow the user of the analysis to either have confidence that there are no errors at runtime if the analysis finds none (if it is sound) or that any errors the analysis finds will actually occur in some runtime scenario (if it is complete). For the purposes of these definitions, an error is a dynamic interpretation of the constraint which causes the requires predicate to fail. In the formal semantics, an error is signaled as a failure for the flow function to produce a new lattice for a particular instruction.

We define soundness and completeness of the relation analysis by assuming an alias analysis which abstracts the heap using $\mathcal{A}$, as described above. For both of these theorems, we let $\mathcal{A}^{\text{conc}}$

Table 1: Differences between sound, complete, and compromise variant

	Trigger Predicate checks when...	Requires Predicate passes when...
Sound	True or Unknown	True
Complete	True	True or Unknown
Compromise	True	True

define the actual heap at some point of an real execution, and we let $\mathcal{A}^{abs}$ be a sound approximation of $\mathcal{A}^{conc}$. We also let ρ_{abs} and ρ_{conc} be relationship lattices consistent with $\mathcal{A}^{abs}$ and $\mathcal{A}^{conc}$ where ρ_{abs} is an abstraction of the concrete runtime lattice ρ_{conc}, defined as $\rho_{conc} \sqsubseteq \rho_{abs}$.

If the relation analysis is sound, we expect that if the flow function runs to completion using the imprecise lattice ρ^{abs}, then any more concrete lattice will also run to completion for that instruction. As the flow function only runs to completion if it finds no errors, then there may be false positives from when ρ^{abs} produces errors, but there will be no false negatives. To be locally sound for this instruction, the analysis must also produce a new abstract lattice that conservatively approximates any new concrete lattice. Theorem 3.1 captures the intuition of local soundness formally. Global soundness follows from local soundness, the monotonicity of the flow function, and the initial conditions of the lattice.

Theorem 3.1 (Local Soundness of Relations Analysis)**.**

if $f_{\mathcal{C};\mathcal{A}^{abs};\mathcal{B}}(\rho^{abs}, instr) = \rho^{abs\prime}$ and $\rho^{conc} \sqsubseteq \rho^{abs}$
then $f_{\mathcal{C};\mathcal{A}^{conc};\mathcal{B}}(\rho^{conc}, instr) = \rho^{conc\prime}$ and $\rho^{conc\prime} \sqsubseteq \rho^{abs\prime}$

If the relation analysis is complete, we expect a theorem which is the opposite of the soundness theorem and is shown in Theorem 3.2. If a flow function runs to completion on a lattice ρ^{conc}, then it will also run to completion on any abstraction of that lattice. An analysis with this property may produce false negatives, as the analysis can find an error using the concrete lattice yet run to completion on the abstract lattice, but it will produce no false positives. Like the sound analysis, the results from the flow function must maintain their existing precision relationship.

Theorem 3.2 (Local Completeness of Relations Analysis)**.**

if $f_{\mathcal{C};\mathcal{A}^{conc};\mathcal{B}}(\rho^{conc}, instr) = \rho^{conc\prime}$ and $\rho^{conc} \sqsubseteq \rho^{abs}$
then $f_{\mathcal{C};\mathcal{A}^{abs};\mathcal{B}}(\rho^{abs}, instr) = \rho^{abs\prime}$ and $\rho^{conc\prime} \sqsubseteq \rho^{abs\prime}$

The relation analysis can be either sound, complete, or a compromise of the two, by making only minor changes to the analysis. Proofs of soundness and completeness, for the sound and complete variants respectively, can be found in the appendicies. The differences between the variants are summarized in Table 1 and are described below.

Trigger condition. The trigger predicate determines when the constraint will check the required predicate and when it will produce effects. The sound analysis will trigger a constraint whenever there is even a possibility of it triggering at runtime. Therefore, it triggers when the predicate is either true or unknown. The complete variant can produce no false positives, so it will only check the requires predicate when the trigger predicate is definitely true. Regardless of the variant, if the trigger is either true or unknown, the analysis produces a set of changes to make to the lattice based upon the effects list.

```
public class ListItemCollection {
    @Item({*, this}, REMOVE)
    public void clear() {...}
    ...
}
```

```
@Constraint(
    op = "ListItemCollection.clear()",
    trigger = "x instanceof ListItem",
    requires = "true",
    effect = {"!Item(x, this)"}
)
```

Figure 4: Translating a relation effect with wildcards into a constraint

Error condition. The requires predicate should be true to signal that the operation is safe to use. The sound variant will cause an error whenever the required predicate is false or unknown. The complete variant, however, can only cause an error if it is sure there is one, so it only flags an error if the requires predicate is definitely false.

Table 1 also shows a variant of the analysis that, while neither sound or complete, we believe is a good compromise between the two. The compromise variant attempts to minimize the number of false positives and false negatives by only triggering when the trigger predicate is definitely true, but then signaling an error if the requires predicate is either false or unknown. While this version can produce false positives and false negatives, we believe it will be the most cost-effective compromise in practice, based on our experience described in Section 5. Additionally, this version may utilize inferred relations, a feature which is inherently neither sound or complete, but reduces the specification burden on the framework developer.

4 Abstract Semantics

In this section, we present formal semantics for a simplified version of the specifications and analysis, the grammar for which is shown in Figure 5. We do not specialized relations for equality(==) and typing (instanceof). It is possible to add specialized relations by calling out to other flow analyses in the same manner as is done with both the boolean constant propagation analysis and the alias analysis.

Relation effects and wildcards are both syntactic sugar that can be easily translated into a constraint form. Relation effects are translated by considering them as a constraint on the annotated method with a `true` trigger predicate, a `true` requires predicate, and the effect list as annotated. Wildcards are easily rewritten by declaring a fresh variable in the trigger predicate and constraining it to have the desired type. Figure 4 shows an example effect with a wildcard translated into a constraint.

The lattice ρ has the usual operators of join ($\sqcup$) and precision ($\sqsubseteq$), which work as expected for a tuple lattice. We also introduce three additional operators, defined in Figure 6. Equivalence join ($\sqcup_{=}$) will resolve to unknown if the two sides are not equal. Overriding meet ($\sqcap_{\leftarrow}$) has the property that if the right side has a defined value (not `bot`), then it will use the right value, otherwise it will use the left value. The polarity operator ($\updownarrow$) will push all non-bottom values to the top of the lattice. Finally, we also define $\bot_{\mathcal{A}}$ as a special lattice which is consistent with the alias lattice $\mathcal{A}$ and which maps every relationship to `bot`.

constraint	$cons$	::=	$op : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q}$
predicate	P	::=	$P_1 \wedge P_2 \mid P_1 \vee P_2 \mid P_1 \implies P_2 \mid Q \mid true \mid false$
negation predicate	Q	::=	$\neg S \mid S$
test predicate	S	::=	$A \mid A/y$
relation predicate	A	::=	$\mathsf{rel}(\bar{y})$
bound predicate	M	::=	$M_1 \wedge M_2 \mid M_1 \vee M_2 \mid M_1 \implies M_2 \mid N \mid true \mid false$
bound negation	N	::=	$\neg T \mid T$
bound test	T	::=	$R \mid R/\ell$
relationship	R	::=	$\mathsf{rel}(\bar{\ell})$
source instruction	$instr$	::=	$\mathtt{x_{ret}} = \mathtt{x_{this}}.\mathsf{m}(\bar{\mathtt{x}}) \mid \mathtt{x_{ret}} = \mathtt{new}\ \tau(\bar{\mathtt{x}}) \mid eom \mid \ldots$
instruction signature	op	::=	$\tau_{\mathtt{this}}.\mathsf{m}(\bar{y} : \bar{\tau}) : \tau_{\mathtt{ret}} \mid \mathsf{new}\ \tau(\bar{y} : \bar{\tau}) \mid \mathsf{end\text{-}of\text{-}method} \mid \ldots$
ternary logic	t	::=	$\mathsf{True} \mid \mathsf{False} \mid \mathsf{Unknown}$
lattice elements	E	::=	$\mathtt{unknown} \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{bot}$
flow lattice	ρ	::=	$R \mapsto E, \rho \mid \varnothing$
set of lattices	$\mathcal{P}$	::=	$\{\rho\} \cup \mathcal{P} \mid \varnothing$
substitution	σ	::=	$(y \mapsto \ell), \sigma \mid \varnothing$
set of substitutions	Σ	::=	$\{\sigma\} \cup \Sigma \mid \varnothing$
bool constants lattice	$\mathcal{B}$	::=	$\ell \mapsto t, \mathcal{B} \mid \varnothing$
alias lattice	$\mathcal{A}$	::=	$< \Gamma_\ell; \mathcal{L} >$
aliases	$\mathcal{L}$	::=	$(\mathtt{x} \mapsto \bar{\ell}), \mathcal{L} \mid \varnothing$
location types	Γ_ℓ	::=	$(\ell : \tau), \Gamma_\ell \mid \varnothing$
spec variable types	Γ_y	::=	$(y : \tau), \Gamma_y \mid \varnothing$
relation type	$\mathcal{R}$	::=	$\mathsf{rel} \mapsto \bar{\tau}, \mathcal{R} \mid \varnothing$
constraints	$\mathcal{C}$	::=	$cons, \mathcal{C} \mid \varnothing$
relation inference rules	$\mathcal{I}$	::=	$P \Downarrow \bar{S}, \mathcal{I} \mid \varnothing$

$\mathtt{x}$ is a source variable
m is a method name
rel is a relation name
τ is a type
y is a spec variable, where the variables this and ret have special meanings
ℓ is a label for a runtime object

$\bot_{\mathcal{A}}$ is a special lattice which is consistent with the alias lattice $\mathcal{A}$ and where every relationship maps to $\mathtt{bot}$

Figure 5: Abstract grammar

4.1 Checking predicate truth

Before we show how constraint checking works, we must describe how the analysis tests the truth of a relationship predicate. The judgment for this is written as

$$\mathcal{A}; \mathcal{B}; \rho \vdash M\ t$$

and is read as "Given an aliasing context and a constant propagation context, the lattice ρ shows that bound predicate M is t", where t is either True, False, or $\mathsf{Unknown}$. The rules for this judgment

$$\frac{}{\mathsf{E} \mathbin{\boxed{\leftarrow}} \mathtt{bot} = \mathsf{E}}\,(\textsc{ovrmeet-bot}) \qquad \frac{\mathsf{E}_r \neq \mathtt{bot}}{\mathsf{E}_l \mathbin{\boxed{\leftarrow}} \mathsf{E}_r = \mathsf{E}_r}\,(\textsc{ovrmeet-not-bot})$$

$$\frac{}{\mathsf{E} \mathbin{\boxed{=}} \mathsf{E} = \mathsf{E}}\,(\textsc{eqjoin}{-}{=}) \qquad \frac{\mathsf{E}_l \neq \mathsf{E}_r}{\mathsf{E}_l \mathbin{\boxed{=}} \mathsf{E}_r = \mathsf{unknown}}\,(\textsc{eqjoin}{-}{\neq})$$

$$\frac{}{\updownarrow \mathtt{bot} = \mathtt{bot}}\,(\textsc{polar-bot}) \qquad \frac{\mathsf{E} \neq \mathtt{bot}}{\updownarrow \mathsf{E} = \mathsf{unknown}}\,(\textsc{polar-unknown})$$

$$\frac{}{\mathtt{bot} \trianglelefteq \mathtt{bot}}\,(\trianglelefteq\textsc{-bot}) \qquad \frac{}{\mathtt{bot} \trianglelefteq \mathsf{unknown}}\,(\trianglelefteq\textsc{-unknown}) \qquad \frac{\mathsf{E}_l \neq \mathtt{bot}}{\mathsf{E}_l \trianglelefteq \mathsf{E}_r}\,(\trianglelefteq\textsc{-other})$$

$$\frac{}{\mathtt{bot} \sqcup \mathsf{E} = \mathsf{E}}\,(\sqcup\textsc{-bot-l}) \qquad \frac{}{\mathsf{E} \sqcup \mathtt{bot} = \mathsf{E}}\,(\sqcup\textsc{-bot-r}) \qquad \frac{}{\mathsf{E} \sqcup \mathsf{E} = \mathsf{E}}\,(\sqcup{-}{=})$$

$$\frac{\mathsf{E}_l \neq \mathtt{bot} \qquad \mathsf{E}_r \neq \mathtt{bot} \qquad \mathsf{E}_l \neq \mathsf{E}_r}{\mathsf{E}_l \sqcup \mathsf{E}_r = \mathsf{unknown}}\,(\sqcup{-}{\neq})$$

$$\frac{}{\mathtt{bot} \sqsubseteq \mathsf{E}}\,(\sqsubseteq\textsc{-bot}) \qquad \frac{}{\mathsf{E} \sqsubseteq \mathsf{unknown}}\,(\sqsubseteq\textsc{-unknown}) \qquad \frac{\mathsf{E} \neq \mathtt{bot} \qquad \mathsf{E} \neq \mathsf{unknown}}{\mathsf{E} \sqsubseteq \mathsf{E}}\,(\sqsubseteq{-}{=})$$

Figure 6: Lattice Element Operations

are similar to three-valued logic and are shown in Figures 7 and 8.

In the sound and complete variants, the rules are trivial. The analysis inspects the lattice to see what the value of the relationship is to determine whether it is True (REL-T), False (REL-F), or Unknown (REL-U-SOUND/COMPLETE). If the lattice maps the relationship to either `unknown` or `bot`, then the predicate is considered Unknown. The rest of the predicate rules work as expected for a three-valued logic.

The interesting case is in the compromise variant when the relationship does not map to `true` or `false`. Instead of using the rule (REL-U-SOUND/COMPLETE), the compromise variant admits the rules (REL-U-COMPROMISE) and (INFER-COMPROMISE). These rules attempt to use the inferred relationships, defined in Section 2.3, to retrieve the desired relationship. The rule for the inference judgement ρ infers ρ', is defined in Figure 9. This rule first checks to see if the trigger of an inferred relation is true, and if so, uses the function `lattice` to produce the inferred relationships described by $\bar{R}[\sigma]$. For all relationships not defined by $R[\bar{\sigma}]$, the lattice function defaults to `bot` to signal that there are no changes. There are two properties to note about the rules (REL-U-COMPROMISE), (INFER-COMPROMISE), and (DISCOVER):

1. The use of inferred relationships does not change the original lattice ρ. This allows the inferred relationships to go away automatically if the generating predicate, P, is no longer true.
2. Any inferred relationship must be *strictly more precise* than the relationship's value in ρ, as enforced by $\rho' \sqsubset \rho$. This means that relationships can move from `unknown` to `true`, but they can not move from `false` to `true`. This property guarantees termination and prevents the inferred relationships from taking precedence over declared ones.

Inferred relationships can not be used in the sound and complete variants. This does not limit

$$\boxed{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}\ \mathsf{t}}$$

$$\frac{\rho(\mathsf{R}) = \texttt{true}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{True}}\,(\text{REL}-\text{TRUE}) \qquad \frac{\rho(\mathsf{R}) = \texttt{false}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{False}}\,(\text{REL}-\text{FALSE})$$

$$\frac{\rho(\mathsf{R}) = \mathsf{E} \qquad \mathsf{E} \neq \texttt{true} \qquad \mathsf{E} \neq \texttt{false}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{Unknown}}\,(\text{REL}-\text{UNKNOWN}-\text{SOUND/COMPLETE})$$

$$\frac{\begin{array}{c}\rho(\mathsf{R}) = \mathsf{E} \qquad \mathsf{E} \neq \texttt{true} \qquad \mathsf{E} \neq \texttt{false} \\ \mathcal{A};\mathcal{B} \vdash \rho\ \mathsf{infers}\ \rho' \qquad \rho \mathrel{\boxed{\scriptstyle\leftarrow}} \rho' \vdash \mathsf{R}\ \mathsf{t} \qquad \mathsf{t}\ \mathsf{is}\ \mathsf{True}\ \mathsf{or}\ \mathsf{False}\end{array}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{t}}\,(\text{INFER}-\text{COMPROMISE})$$

$$\frac{\begin{array}{c}\rho(\mathsf{R}) = \mathsf{E} \qquad \mathsf{E} \neq \texttt{true} \qquad \mathsf{E} \neq \texttt{false} \\ \neg\exists\rho\ .\ \mathcal{A};\mathcal{B} \vdash \rho\ \mathsf{infers}\ \rho'\end{array}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{Unknown}}\,(\text{REL}-\text{UNKNOWN}-\text{COMPROMISE})$$

$$\frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{t} \qquad \mathcal{B}(\ell_{\mathtt{test}}) = \mathsf{t} \qquad \mathsf{t} \neq \mathsf{Unknown}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}/\ell_{\mathtt{test}}\ \mathsf{True}}\,(\text{REL}-\text{TEST}-\text{T})$$

$$\frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{t}_1 \qquad \mathcal{B}(\ell_{\mathtt{test}}) = \mathsf{t}_2 \qquad \mathsf{t}_1 \neq \mathsf{Unknown} \qquad \mathsf{t}_2 \neq \mathsf{Unknown} \qquad \mathsf{t}_1 \neq \mathsf{t}_2}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}/\ell_{\mathtt{test}}\ \mathsf{False}}\,(\text{REL}-\text{TEST}-\text{F})$$

$$\frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{Unknown}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}/\ell_{\mathtt{test}}\ \mathsf{Unknown}}\,(\text{REL}-\text{TEST}-\text{U1}) \qquad \frac{\mathcal{A};\mathcal{B}(\ell_{\mathtt{test}}) = \ \mathsf{Unknown} \qquad \mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}\ \mathsf{t}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{R}/\ell_{\mathtt{test}}\ \mathsf{Unknown}}\,(\text{REL}-\text{TEST}-\text{U2})$$

$$\frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{T}\ \mathsf{Unknown}}{\mathcal{A};\mathcal{B};\rho \vdash \neg\mathsf{T}\ \mathsf{Unknown}}\,(\neg\text{R}-\text{UNKNOWN}) \qquad \frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{T}\ \mathsf{False}}{\mathcal{A};\mathcal{B};\rho \vdash \neg\mathsf{T}\ \mathsf{True}}\,(\neg\text{R}-\text{TRUE}) \qquad \frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{T}\ \mathsf{True}}{\mathcal{A};\mathcal{B};\rho \vdash \neg\mathsf{T}\ \mathsf{False}}\,(\neg\text{R}-\text{FALSE})$$

$$\frac{}{\mathcal{A};\mathcal{B};\rho \vdash \mathtt{true}\ \mathsf{True}}\,(\text{TRUE}) \qquad \frac{}{\mathcal{A};\mathcal{B};\rho \vdash \mathtt{false}\ \mathsf{False}}\,(\text{FALSE}) \qquad \frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}_1\ \mathsf{False}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}_1 \implies \mathsf{M}_2\ \mathsf{True}}\,(\implies -\text{TRUE1})$$

$$\frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{P}_2\ \mathsf{True}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}_1 \implies \mathsf{M}_2\ \mathsf{True}}\,(\implies -\text{TRUE2}) \qquad \frac{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}_1\ \mathsf{True} \qquad \mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}_2\ \mathsf{False}}{\mathcal{A};\mathcal{B};\rho \vdash \mathsf{M}_1 \implies \mathsf{M}_2\ \mathsf{False}}\,(\implies -\text{FALSE})$$

Figure 7: Check predicate truth under a lattice

the expressiveness of the specifications, as inferred relations can always be written directly within the constraints. Doing so does make the specifications more difficult to write; the framework developer must add the inferred relations to any constraint which will also prove the trigger predicate. Since inferred relations do change the semantics, they are not syntactic sugar, but they are not necessary for reasons beyond the ease of writing specifications.

4.2 Matching on an operator

In order to check a constraint, the analysis must determine whether a source instruction, called $\mathtt{instr}$, matches the syntactic operation op defined by a constraint. This is realized in the judgment

$$\mathcal{A};\Gamma_y \vdash \mathtt{instr} : \mathsf{op} \Mapsto (\Sigma^t, \Sigma^u)$$

$$\boxed{\mathcal{A}; \mathcal{B}; \rho \vdash M\ t}$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ Unknown} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ Unknown}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \implies M_2 \text{ Unknown}}\,(\implies -\text{UNKNOWN1})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ True} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ Unknown}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \implies M_2 \text{ Unknown}}\,(\implies -\text{UNKNOWN2})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ Unknown} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ False}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \implies M_2 \text{ Unknown}}\,(\implies -\text{UNKNOWN3})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ True} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ True}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \wedge M_2 \text{ True}}\,(\wedge-\text{TRUE}) \qquad \frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{False}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \wedge M_2 \text{False}}\,(\wedge-\text{FALSE1})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ False}}{\mathcal{B}; \rho \vdash M_1 \wedge M_2 \text{ False}}\,(\wedge-\text{FALSE2}) \qquad \frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ True} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ Unknown}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \wedge M_2 \text{ Unknown}}\,(\wedge-\text{UNKNOWN1})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ Unknown} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ True}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \wedge M_2 \text{ Unknown}}\,(\wedge-\text{UNKNOWN2})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ Unknown} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ Unknown}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \wedge M_2 \text{ Unknown}}\,(\wedge-\text{UNKNOWN3})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ True}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \vee M_2 \text{ True}}\,(\vee-\text{TRUE1}) \qquad \frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ True}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \vee M_2 \text{ True}}\,(\vee-\text{TRUE2})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ False} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ False}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \vee M_2 \text{ False}}\,(\vee-\text{FALSE})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ False} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ Unknown}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \vee M_2 \text{ Unknown}}\,(\vee-\text{UNKNOWN1})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ Unknown} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ False}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \vee M_2 \text{ Unknown}}\,(\vee-\text{UNKNOWN2})$$

$$\frac{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \text{ Unknown} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash M_2 \text{ Unknown}}{\mathcal{A}; \mathcal{B}; \rho \vdash M_1 \vee M_2 \text{ Unknown}}\,(\vee-\text{UNKNOWN3})$$

Figure 8: Check predicate truth under a lattice

$$\boxed{\rho \text{ infers } \rho'}$$

$$\frac{P \Downarrow \overline{Q} \in \mathcal{I} \qquad \rho \vdash P[\sigma] \text{ True} \qquad \rho' = \mathtt{lattice}(\overline{Q}[\sigma]; \mathcal{A}; \mathcal{B}) \qquad \rho' \sqsubset \rho}{\mathcal{A}; \mathcal{B} \vdash \rho \text{ infers } \rho'}\,(\text{DISCOVER})$$

Figure 9: Infer new relationships

with rules defined in Figure 10. Given the alias lattice $\mathcal{A}$ and a typing environment for the free variables in $\mathtt{op}$, this judgment matches $\mathtt{instr}$ to $\mathtt{op}$ and produces two disjoint sets of substitutions

$$\boxed{\mathcal{A};\Gamma_y \vdash instr : \text{op} \Mapsto (\Sigma^t,\Sigma^u)}$$

$$\frac{FV(\tau_{this}.\text{m}(\overline{\text{y}}:\overline{\tau}):\tau_{ret}) \subseteq \Gamma_y \qquad (\Sigma^t,\Sigma^u) = \text{findLabels}(\mathcal{A};\Gamma_y;\mathtt{x_{ret}},\mathtt{x_{this}},\overline{\mathtt{x}};\text{ret},\text{this},\overline{\text{y}})}{\mathcal{A};\Gamma_y \vdash \mathtt{x_{ret}} = \mathtt{x_{this}}.\text{m}(\overline{\mathtt{x}}) : \tau_{this}.\text{m}(\overline{\text{y}}:\overline{\tau}):\tau_{ret} \Mapsto (\Sigma^t,\Sigma^u)}\;\text{(INVOKE)}$$

$$\frac{\begin{array}{c} FV(\text{ new }\ \tau(\overline{\text{y}}:\overline{\tau})) \subseteq \Gamma_y \\ (\Sigma^t,\Sigma^u) = \text{findLabels}(\mathcal{A};\Gamma_y;\mathtt{x_{ret}},\overline{\mathtt{x}};\text{this},\overline{\text{y}}) \end{array}}{\mathcal{A};\Gamma_y \vdash \mathtt{x_{ret}} = \mathtt{new}\ m(\overline{\mathtt{x}}) :\ \text{new}\ \ \tau(\overline{\text{y}}:\overline{\tau}) \Mapsto (\Sigma^t,\Sigma^u)}\;\text{(CONSTRUCTOR)}$$

$$\frac{}{\mathcal{A};\Gamma_y \vdash eom : \text{end-of-method} \Mapsto (\{\varnothing\},\varnothing)}\;\text{(EOM)}$$

$$\boxed{\text{findLabels}(\mathcal{A};\Gamma_y;\overline{\mathtt{x}};\overline{\text{y}}) = (\Sigma^t,\Sigma^u)}$$

$$\frac{\begin{array}{c} |\overline{\mathtt{x}}| = |\overline{\text{y}}| = n \\ \Sigma^t = \{(\text{y}_1 \mapsto \ell_1),\ldots,(\text{y}_n \mapsto \ell_n) \mid \\ \forall\, i \in 1\ldots n \,.\, \mathcal{L}(\mathtt{x_i}) = \{\ell_i\} \;\wedge\; \Gamma_\ell(\ell_i) <: \Gamma_y(\text{y}_i)\} \\ \Sigma^u = \{(\text{y}_1 \mapsto \ell_1),\ldots,(\text{y}_n \mapsto \ell_n) \mid \\ \forall\, i \in 1\ldots n \,.\, \ell_i \in \mathcal{L}(\mathtt{x_i}) \;\wedge\; \exists\, \tau' \,.\, \tau' <: \Gamma_\ell(\ell_i) \;\wedge\; \tau' <: \Gamma_y(\text{y}_i)\} - \Sigma^t \end{array}}{\text{findLabels}(<\Gamma_\ell,\mathcal{L}>;\Gamma_y;\overline{\mathtt{x}};\overline{\text{y}}) = (\Sigma^t,\Sigma^u)}\;\text{(FINDLABELS)}$$

Figure 10: Matching instructions to operations and type satisfaction

that map specification variables in op to heap locations. The first set, Σ^t, represents possible substitutions where the locations are all known to be a subtype of the type required by the variables. The second set, Σ^u, are potential substitutions where the locations may or may not have the right type at runtime.

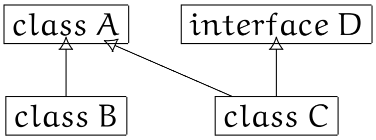
As an example, we will walk through the rule (INVOKE). The first premise checks that the free variables in op are in Γ_y, and the second premise builds the substitution set using the findLabels function. Each substitution in the set will map the specification variables in op (this, ret, and $\text{y}_1 \ldots \text{y}_n$) to a location in the heap that is aliased by the appropriate source variables in $instr$ ($\mathtt{x_{this}}$, $\mathtt{x_{ret}}$, and $\mathtt{x_1} \ldots \mathtt{x_n}$).

To produce the set Σ^t, the findLabels function must generate a substitution for each y_i in $\overline{\text{y}}$. It starts by verifying that the corresponding source variable $\mathtt{x_i}$ points to only one location ℓ, and it checks to see if the type of that location is a subtype of the type required for y_i. Every substitution σ which fits these requirements is in Σ^t.

Σ^u is a more interesting set. Unlike Σ^t, it checks all locations which $\mathtt{x_i}$ aliases and records a possible substitution for each. Additionally, when it checks the type, it allows the location if there is even a *possibility* of it being the right type. As an example, consider the class hierarchy and use of findLabels shown in Figure 11. In the first row, ℓ is definitely substitutable for y, so it is a substitution in Σ^t. In the second row, y can never be substituted by ℓ, so both sets are empty. In the third and fourth rows, ℓ may be substitutable for y (if ℓ has type B or C, respectively), so both substitutions are possibly, but not definitely, allowed and are therefore in Σ^u.

The need for Σ^u may seem surprising, but the rationale behind it is that framework constraints do not always adhere to behavioral subtyping. Consider the `DropDownList` selection constraint being analyzed for the code in Listing 8. Since `list` is of type `ListControl`, the trigger clause

$$\mathsf{findLabels}(<\ell:\tau_\ell, \mathtt{x} \mapsto \{\ell\}>; \mathsf{y}:\tau_\mathsf{y}; \mathtt{x}; \mathsf{y}) = (\Sigma^t, \Sigma^u)$$



τ_ℓ	τ_y	Σ^t	Σ^u
B	A	$\{(y \mapsto \ell)\}$	$\varnothing$
B	D	$\varnothing$	$\varnothing$
A	B	$\varnothing$	$\{(y \mapsto \ell)\}$
A	D	$\varnothing$	$\{(y \mapsto \ell)\}$

Figure 11: The difference between Σ^t and Σ^u

Listing 8: Generically changing the selection on a `ListControl`

```java
ListControl list = ...;
ListItem item;
item = list.getItems().findByValue("foo");
item.setSelected(true);
```

of the first constraint in Listing 7 will not be true, and the constraint will never trigger an error. However, we would like this to trigger a potential violation in the sound variant since `list` could be a `DropDownList`. The root of the problem was that `DropDownList` is not following the principle of behavioral subtyping; it has added preconditions to methods that the base class did not require. Therefore, a `DropDownList` is not always substitutable where a `ListControl` is used! While frustrating, this appears to be a common problem with frameworks. Inheritance was used here rather than composition because the type is structurally the same, and it is almost behaviorally the same. In fact, the methods on `DropDownList` itself do appear to be behaviorally the same. However, the subtype added a few constraints to *other* classes, like the `ListItem` class.

By keeping track of Σ^t and Σ^u separately, it will allow the variants of the analysis to use them differently. In particular, the sound variant will trigger errors from substitutions in Σ^u, while the complete and compromise variant will only use it to propagate lattice changes from the effect list.

4.3 Checking a single constraint

We will now show how the analysis checks an instruction for a single constraint. This is done with the judgment

$$\mathcal{A}; \mathcal{B}; \rho; cons \vdash instr \hookrightarrow \rho^\Delta$$

shown in Figure 12. This judgment takes the alias lattice, the relationship lattice, and a constraint, and it determines what changes to make to the lattice for the given instruction. The lattice changes are represented in ρ^Δ, where a relationship mapped to `bot` signifies no changes.

The analysis starts by checking whether the instruction matches the operation used by the constraint. If not, then instruction matching rules will return no substitutions, the rule (NO-MATCH) will apply, and no changes are made by returning $\bot_\mathcal{A}$. If there are substitutions, as shown in rule (MATCH), then the analysis must check this constraint for every aliasing configuration possible, as represented by Σ^t and Σ^u. This rule checks that the constraint passes for each aliasing configuration σ and receives the lattice changes for each. If the substitution was from Σ^u, then the analysis must

$$\boxed{\mathcal{A};\mathcal{B};\rho;\mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \rho^{\Delta}}$$

$$\frac{\begin{array}{c}\mathsf{cons} = \mathsf{op} : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \qquad \mathcal{A};\mathsf{FV}(\mathsf{cons}) \vdash \mathsf{instr} : \mathsf{op} \Mapsto (\Sigma^{t}, \Sigma^{u}) \\ \mathcal{P}^{t} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{t} \ \wedge\ \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\ \mathcal{P}^{u} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{u} \ \wedge\ \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\ \Sigma^{t} \neq \varnothing \ \vee\ \Sigma^{u} \neq \varnothing \qquad |\mathcal{P}^{t}| = |\Sigma^{t}| \qquad |\mathcal{P}^{u}| = |\Sigma^{u}| \qquad \mathcal{P}^{\Delta} = \mathcal{P}^{t} \cup \mathcal{P}^{u}\end{array}}{\mathcal{A};\mathcal{B};\rho;\mathsf{cons} \vdash \mathsf{instr} \hookrightarrow (\ \sqcup\ \mathcal{P}^{\Delta})}\ (\mathrm{MATCH})$$

$$\frac{\mathsf{cons} = \mathsf{op} : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \qquad \mathcal{A};\mathsf{FV}(\mathsf{cons}) \vdash \mathsf{instr} : \mathsf{op} \Mapsto (\varnothing, \varnothing)}{\mathcal{A};\mathcal{B};\rho;\mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \bot_{\mathcal{A}}}\ (\mathrm{NO{-}MATCH})$$

$$\boxed{\mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}}$$

$$\frac{\begin{array}{c}\mathsf{cons} = \mathsf{op} : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \\ \Gamma_{y} = \mathsf{FV}(\mathsf{op}) \cup \mathsf{FV}(P_{ctx}) \cup \mathsf{FV}(\overline{Q}) \qquad \mathsf{allValidSubs}(\mathcal{A};\sigma_{op};\Gamma_{y}) = (\Sigma^{t}, \Sigma^{u}) \\ \mathcal{P}^{t} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{t} \ \wedge\ \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{full}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\ \mathcal{P}^{u} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{u} \ \wedge\ \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{full}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\ \Sigma^{t} \neq \varnothing \ \vee\ \Sigma^{u} \neq \varnothing \qquad |\Sigma^{t}| = |\mathcal{P}^{t}| \qquad |\Sigma^{u}| = |\mathcal{P}^{u}| \qquad \mathcal{P}^{\Delta} = \mathcal{P}^{t} \cup \mathcal{P}^{u}\end{array}}{\mathcal{A};\mathcal{B};\rho;\sigma_{op} \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow (\ \sqcup\ \mathcal{P}^{\Delta})}\ (\mathrm{BOUND})$$

$$\frac{\begin{array}{c}\mathsf{cons} = \mathsf{op} : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \\ \Gamma_{y} = \mathsf{FV}(\mathsf{op}) \cup \mathsf{FV}(P_{ctx}) \cup \mathsf{FV}(\overline{Q}) \qquad \mathsf{allValidSubs}(\mathcal{A};\sigma_{op};\Gamma_{y}) = (\varnothing, \varnothing)\end{array}}{\mathcal{A};\mathcal{B};\rho;\sigma_{op} \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \bot_{\mathcal{A}}}\ (\mathrm{CANT{-}BIND})$$

$$\boxed{\mathsf{allValidSubs}(\mathcal{A};\sigma;\Gamma_{y}) = (\Sigma^{t}, \Sigma^{u})}$$

$$\frac{\begin{array}{c}\Sigma^{t} = \{\sigma' \mid \sigma' \supseteq \sigma \ \wedge\ \mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_{y}) \ \wedge\ \forall\, y \mapsto \ell \in \sigma' \,.\, \Gamma_{\ell}(\ell) <: \Gamma_{y}(y)\} \\ \Sigma^{u} = \{\sigma' \mid \sigma' \supseteq \sigma \ \wedge\ \mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_{y}) \ \wedge \\ \forall\, y \mapsto \ell \in \sigma' \,.\, \exists\, \tau' \,.\, \tau' <: \Gamma_{\ell}(\ell) \ \wedge\ \tau' <: \Gamma_{y}(y)\} - \Sigma^{t}\end{array}}{\mathsf{allValidSubs}(<\Gamma_{\ell};\mathcal{L}>;\sigma;\Gamma_{y}) = (\Sigma^{t}, \Sigma^{u})}\ (\mathrm{VALIDSUBS})$$

Figure 12: Check a single constraint

use the $\updownarrow$ operator on the change lattice and the starting lattice to produce the correct change lattice. This is done because the analysis cannot be sure if the substitution is valid at runtime, so it can only make changes into the unknown state. Setting all changes to `unknown` could cause the analysis to lose precision when ρ^{Δ} prescribes a change that already exists in ρ. A possible solution is to let the polarizing operator return `bot` if the prescribed changes already exist in the lattice ρ (we have not yet proven this extension is sound).

The last step the rule makes is to combine all the lattice changes, from all substitutions, using $\sqcup$. The use of $\sqcup$ means that a change is only made to `true` or `false` if all the aliasing configurations agree to it. Likewise, a signal to make no changes by way of `bot` must also show in all configurations. If any configurations disagree about a lattice change, then the lattice element changes to `unknown`.

Once the analysis has a syntactic match, it tries to find the aliasing configurations for a semantic match using the judgment

$$\mathcal{A}; \rho; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^{\Delta}$$

The analysis must get all aliasing configurations that are consistent with the current aliases in σ and Γ_{y}. σ represents the substitutions which are already made by matching the instruction, while Γ_{y} represents the free variables and their types which the analysis should find substitutions for. The substitutions are found by the $\mathsf{allValidSubs}$ function, shown in Figure 12. The rule (BOUND) proceeds in a similar manner to the rule (MATCH), except it checks the constraint using the judgment

$$\mathcal{A}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}$$

The rules for this judgment, shown in Figure 13, are the primary point of difference between the variants of the analysis.

Sound Variant

The sound variant first checks $\mathsf{P}_{\mathsf{trg}}[\sigma]$ under ρ. It uses this to determine which rule applies. If $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is True, as seen in rule (FULL-T-SOUND), then the analysis must check if $\mathsf{P}_{\mathsf{req}}$ is True under ρ given any substitution. Since this is the sound variant, it will only accept substitutions from Σ^{t}. If $\mathsf{P}_{\mathsf{req}}$ is not True with a substitution from Σ^{t}, then the analysis produces an error. If there is no error, the rule produces the effects dictated by $\bar{\mathsf{R}}[\sigma]$. The function $\mathit{lattice}$ simply converts this list to a lattice, where all unspecified relationships map to $\mathtt{bot}$. If $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is False, then the analysis uses rule (FULL-F-SOUND). In this situation the constraint does not trigger, so the requires predicate is not checked and the analysis returns no changes using $\bot_{\mathcal{A}}$.

In the case that $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is $\mathsf{Unknown}$, the sound variant proceeds in a similar manner to the case where $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is True as it must consider the possibility that the trigger predicate is actually true. In fact the only difference in the rule (FULL-U-SOUND) is that the analysis must use the polarizing operator to be conservative with the effects it is producing in case the trigger predicate was actually false.

Complete Variant

Like the sound variant, the complete variant starts by checking $\mathsf{P}_{\mathsf{trg}}[\sigma]$ under ρ. If $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is True, as seen in rule (FULL-T-COMPLETE), then the analysis must check $\mathsf{P}_{\mathsf{req}}$ under ρ given any substitution. As this is the complete variant, the analysis does not care whether the substitution came from Σ^{t} or Σ^{u}, and it does not matter whether $\mathsf{P}_{\mathsf{req}}$ is True or $\mathsf{Unknown}$. If no substitutions work, either because none exist or because they all show $\mathsf{P}_{\mathsf{req}}$ to be false, then the analysis produces an error. Otherwise, if there is no error, then the rule produces some effects. Since the constraint trigger was true, it will produce exactly the effects dictated by $\bar{\mathsf{R}}[\sigma]$. If the analysis determines that $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is False, then it uses the rule (FULL-F-COMPLETE). Like the sound variant, the requires predicate is not checked and the analysis returns no changes.

Finally, if $\mathsf{P}_{\mathsf{trg}}[\sigma]$ is $\mathsf{Unknown}$, the complete variant will not check $\mathsf{P}_{\mathsf{req}}$ as it cannot be sure whether the constraint is actually triggered and it should not produce an error. However, it must still produce some conservative effects in case the constraint is triggered given a more concrete lattice. Like the sound rule in the case of an unknown trigger, the rule uses the polarizing operator $\updownarrow$ to produce only conservative effects.

$$\boxed{\mathcal{A}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \rho^{\Delta}, \text{Sound Variant}}$$

$$\frac{\begin{array}{c}\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{True} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathtt{cons})) \\ \exists\, \sigma' \in \Sigma^{t}\ .\ \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{req}}[\sigma']\ \mathsf{True}\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \mathtt{lattice}(\bar{\mathsf{Q}}[\sigma]; \mathcal{A}; \mathcal{B})}\ \text{(FULL−T−SOUND)}$$

$$\frac{\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{False}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \bot_{\mathcal{A}}}\ \text{(FULL−F−SOUND)}$$

$$\frac{\begin{array}{c}\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{Unknown} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathtt{cons})) \\ \exists\, \sigma' \in \Sigma^{t}\ .\ \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{req}}[\sigma']\ \mathsf{True} \qquad \rho^{\Delta} = \mathtt{lattice}(\bar{\mathsf{Q}}[\sigma]; \mathcal{A}; \mathcal{B})\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow\!\updownarrow \rho^{\Delta}}\ \text{(FULL−U−SOUND)}$$

$$\boxed{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \rho^{\Delta}, \text{Complete Variant}}$$

$$\frac{\begin{array}{c}\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{True} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathtt{cons})) \\ \exists\, \sigma' \in \Sigma^{t} \cup \Sigma^{u}\ .\ \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{req}}[\sigma']\ \mathsf{True} \vee \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{req}}[\sigma']\ \mathsf{Unknown}\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \mathtt{lattice}(\bar{\mathsf{Q}}[\sigma]; \mathcal{A}; \mathcal{B})}\ \text{(FULL−T−COMPLETE)}$$

$$\frac{\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{False}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \bot_{\mathcal{A}}}\ \text{(FULL−F−COMPLETE)}$$

$$\frac{\begin{array}{c}\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \\ \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{Unknown} \qquad \rho^{\Delta} = \mathtt{lattice}(\bar{\mathsf{Q}}[\sigma]; \mathcal{A}; \mathcal{B})\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow\!\updownarrow \rho^{\Delta}}\ \text{(FULL−U−COMPLETE)}$$

$$\boxed{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \rho^{\Delta}, \text{Compromise Variant}}$$

$$\frac{\begin{array}{c}\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{True} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathtt{cons})) \\ \exists\, \sigma' \in \Sigma^{t}\ .\ \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{req}}[\sigma']\ \mathsf{True}\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \mathtt{lattice}(\bar{\mathsf{Q}}[\sigma]; \mathcal{A}; \mathcal{B})}\ \text{(FULL−T−COMPROMISE)}$$

$$\frac{\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{False}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \bot_{\mathcal{A}}}\ \text{(FULL−F−COMPROMISE)}$$

$$\frac{\begin{array}{c}\mathtt{cons} = \mathtt{op} : \mathsf{P}_{\mathtt{ctx}} \Rightarrow \mathsf{P}_{\mathtt{req}} \Downarrow \overline{\mathsf{Q}} \\ \mathcal{A}; \mathcal{B}; \rho \vdash \mathsf{P}_{\mathtt{ctx}}[\sigma]\ \mathsf{Unknown} \qquad \rho^{\Delta} = \mathtt{lattice}(\bar{\mathsf{Q}}[\sigma]; \mathcal{A}; \mathcal{B})\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow\!\updownarrow \rho^{\Delta}}\ \text{(FULL−U−COMPROMISE)}$$

Figure 13: Checking a fully bound constraint and producing effects. Shading highlights the differences between the three variants.

$$\boxed{f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathtt{instr}) = \rho'}$$

$$\frac{\begin{array}{c} f_{\mathtt{alias}}(\mathcal{A}, \mathtt{instr}) = \mathcal{A}' \\ \forall\, \mathtt{cons}_i \in \mathcal{C} \,.\, \mathcal{A}'; \mathcal{B}; \rho; \mathtt{cons}_i \vdash \mathtt{instr} \hookrightarrow \rho_i^{\Delta} \qquad \rho^{\Delta} = \sqcup\{\rho_i^{\Delta}\} \qquad (i \in 1 \ldots n) \end{array}}{f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathtt{instr}) = \mathsf{transfer}(\rho, \mathcal{A}') \mathrel{\overleftarrow{\sqcap}} \rho^{\Delta}}\,(\textsc{flow-cons})$$

Figure 14: The flow function for the relation analysis

Compromise Variant

The compromise variant is a combination of the sound and complete variants. It has the same rule for False as the other two variants, (FULL-F-COMPROMISE). The rule (FULL-T-COMPROMISE) is the same as the True rule for soundness, while the rule (FULL-U-COMPROMISE) is the same as the Unknown rule for completeness. This means that this variant can produce both false positives and false negatives. The false negatives can occur when $\mathsf{P}_{\mathtt{trg}}$ is Unknown under ρ, but a more precise lattice would have found $\mathsf{P}_{\mathtt{trg}}$ to be True and eventually generated an error. The false positives occur when $\mathsf{P}_{\mathtt{trg}}$ is True under ρ and $\mathsf{P}_{\mathtt{req}}$ is Unknown under ρ, but $\mathsf{P}_{\mathtt{req}}$ would have been True under a more precise lattice.

4.4 The flow function

The flow function for the analysis checks all the individual constraints and produces the final lattice for each operation. Using the judgments defined in the previous section, the flow function iterates through each constraint and receives a change lattice for each. As shown in Figure 14, these lattices are combined using the join operator. Once the analysis has the final change lattice ρ^{Δ}, it applies the changes using the overriding meet operation. This will preserve the old values of a relationship if the change lattice maps to `bot`, but it will override the old value otherwise. This provides us with the new relationship lattice ρ', which is used by the dataflow analysis to feed into the next instruction's flow function. This flow function is monotonic, and the lattice has a finite height, so the dataflow analysis will reach a fix point.

5 Implementation and Experience

We implemented the compromise variant of the analysis in the Crystal dataflow analysis framework, an Eclipse plugin developed at Carnegie Mellon University for statically analyzing Java source [4]. Crystal provides capabilities for analyzing source in three address code form, running a branch-sensitive analysis, and reading specifications from annotations. For the implementation of this analysis, we also used a boolean constant propagation analysis and a basic alias analysis. Either of these could be replaced with more sophisticated analyses in order to improve the results; the relation analysis is only dependent on the interfaces to these analyses.

We specified three constraints, one from the ASP.NET framework[5] and two from the Eclipse

[4] `http://code.google.com/p/crystalsaf`

[5] We translated the relevant parts of the API and the examples into Java.

JDT framework. These were all constraints which we had misused ourselves and were common problems that were posted on the help forums and mailing lists. These constraints exercised several different patterns, and the specifications were able to capture each of these patterns.

The specifications allowed us to easily describe structured relationships, such as the `ListItem`s which are in a `DropDownList` and a tree of `ASTNode`s within the Eclipse JDT. In each of these cases, a relationship ties the "child" and "parent" objects together, and it is straightforward to check if two children have the same parent. Two of our constraints had a structured relationship where an operation required that some objects exist (or do not exist) in a structured relationship.

All three constraints had semantics which required operations to occur in a particular order. To define this pattern, we just needed a relationship which binds relevant objects together. The operation which occurs first produces an effect which sets this relationship to true, and the operation which must occur second simply requires this relationship. An example of this was seen in the constraints on the `DropDownList` in Listing 7. Additionally, relationships also allowed us to specify partial orderings of operations. One of the Eclipse JDT constraints had this behavior, and in fact required three methods to be called before the constrained operation. Alternatively, the user could choose to call a fourth method that would replace all three method calls. We captured this constraint by having each of the four methods produce a relationship, and the constrained operation simply required either the three relationships produced from the group of three methods, or the single relationship produced from the fourth one.

Relationships also made it straightforward to associate any objects that were used in the same operation. For example, this allowed us to associate several fields of an object so that we could later check that they were only used together. We did this by annotating the constructor of the object with a relationship effect that tied the field parameters together. We could also associate objects that were linked by some secondary object, but had no direct connection, such as a `DropDownList` and the `ListItems` received from calls to the associated `ListItemCollection`.

After specifying the constraints, we ran the compromise analysis on 20 examples based on real-world code. The examples we selected are based on our own misuses of these frameworks and on several postings on internet help forums and mailing lists. Of these, the compromise variant worked properly on 16, meaning that it either found an expected error or did not find an error on correct code. Most of these examples had little aliasing and used exact types, which reflected what we saw on the help forums.

These examples identified two sources of imprecision. The compromise variant failed on one example because it used an unconstrained supertype, and it failed on the remaining three examples because the relevant constraint required objects which were not in scope. The unconstrained supertype resulted in a false negative, and the three examples with objects out of scope resulted in false positives. In all four of these cases, the sound variant would have flagged an error, and the complete variant would not have.

Using an unconstrained supertype, such as using a `ListControl` instead of a `DropDownList`, as seen in Listing 8, is the first potential source of imprecision for the compromise variant. While a sound analysis would have detected the error in this example, in practice, using this superclass is not typical. The plugin has a `DropDownList` as a field if the control was initialized statically on the web page, and the plugin will typically cast directly to the expected subtype if it created the control dynamically. In fact, we never found code on the forum that used the superclass ListControl.

The more interesting, and more typical, source of imprecision occurs when a required object is not in scope. For example, one of the Eclipse JDT constraints required that an ASTNode have a relationship with an AST object. The plugin, however, did not have any AST objects in scope at all, even though this relationship did exist globally. Based on the examples we found, this does occur in practice, typically when the framework makes multiple callbacks in sequence, such as with a Visitor pattern.

Future revisions of the analysis could address the problem of out-of-scope objects with two changes. First, it should be possible for the framework to declare what relationships exist at the point where the callback occurs. This would have provided the correct relationships in the previous example, and it should be relatively straightforward to annotate the interface of the plugin with this information. Second, an inter-procedural analysis on only the plugin code could handle the case where the relationship goes out of scope for similar reasons, such as calls to a helper function. These changes would increase the precision of all three variants of the analysis.

The two sources of imprecision affect all three variants, though in different ways. While imprecision anywhere in the constraint can produce a false positive in the sound variant or a false negative in the complete variant, the location of the imprecision in the constraint directly changes how the compromise variant handles it. When the imprecision occurs in the trigger predicate, the compromise variant results in a false negative. When the trigger predicate is precise but the requires predicate is imprecise, the compromise variant results in a false positive. This reflects what we expect from the analysis; we only wish to see an error if there is reason to believe that the constraint applies to our plugin. If the trigger predicate is unknown, it is less likely that the constraint is relevant.

6 Related Work

SCL [9] allows framework developers to create a specification for the structural constraints for using the framework. The specifications we propose focus on semantic constraints rather than structural constraints. Some of the key ideas from SCL could be used to drive the more structurally focused parts of the specifications, and we view the two as complimentary.

Scoped Methods [16] are a language construct for enforcing protocols which are local to a method, such as a framework callback. Like SCL, scoped methods are structural and do not take semantic context of objects into account.

Typestates [6] provide a mechanism for specifying a protocol on a single object by using a state machine. There have been several approaches to inter-object typestate. Lam et al. manipulated the typestate of many objects together through their participation in data structures [12]. Nanda et al. take this a step further by allowing external objects to affect a particular object's state, but unlike relationships, it requires that the objects reference each other through a pre-defined path [14]. Bierhoff and Aldrich add permissions to typestates and allows objects to capture the permission of another object, thus binding the objects as needed for the protocol [2]. Relationships can combine multiple objects into a single state-like construct and is more general for this purpose than typestate; it can describe all of the examples used in multiple object typestate work.

With respect to the specifications, relationships are more incremental than typestate because the entire protocol does not need to be specified in order to specify a single constraint. Additionally, the plugin developer does not add any specifications, which she must do with some of the typestate approaches. However, typestate analyses aim to be sound, and can also check that both the plugin and the framework meet the specification. The relationship analysis assumes that the framework properly meets the specification and only analyzes the plugin.

Tracematches have also been used to enforce protocols [17]. Unlike typestate, which specifies the correct protocol, tracematches specify a temporal sequence of events which lead to an error state. This is done by defining a state machine for the protocol and then specifying the bad paths.

The tracematch specification approach is similar to that of relationships; the main difference is in how the techniques specify the path leading up to the error state. Tracematches must specify the entire good path leading up to the error state, which leads to many specifications to define a single bad error state. In cases where multiple execution traces lead to the same error, such as the many ways to find an item in a `DropDownList` and select it incorrectly, a tracematch would have to specify each possibility. Instead of specifying the good path leading up to the error, relationships specify the context predicate, which is the same for all good paths. This difference affects how robust a specification is in the face of API changes. If the framework developer adds a new way to access `ListItem`s in a `ListControl`, the existing tracematches will not cover that good path. However, all the constraint specifications in the proposed technique will continue to work if the new method is annotated with the appropriate relationship effects.

Unlike relationships, tracematches are enforced both dynamically and statically using a global analysis [4]. The static analysis soundly determines possible violations, and it instruments the code to check them dynamically. Bodden et al. provide a static analysis which optimizes the dynamic analysis by verifying more errors statically [5], and Naeem and Lhoták specifically optimize with regard to tracematches that involve multiple objects [13] .

Bierman and Wren formalized UML relationships as a first-class language construct [3]. The language extension they created gives relationships attributes and inheritance, and plugin developers use the relationships by explicitly adding and removing them. In contrast, the relationships presented in this paper are added and removed implicitly through use of framework operations, and if inferred relationships are used, they may be entirely hidden from the developer. While Bierman and Wren did not explore constraints on relationships, Balzer et al. discuss how to describe relationship invariants using discrete mathematics [1]. These invariants are on the relationships themselves and, unlike the proposed work, they do not constrain the framework operations.

Like the proposed framework language, Contracts [8] also view relationships between objects as a key factor in specifying systems. A contract also declares the objects involved in the contract, an invariant, and a lifetime where the invariant is guaranteed to hold. Contracts allow all the power of first-order predicate logic and can express very complex invariants. Contracts differ from the proposed specifications because they do not check the conformance of plugins and the specifications are more complex to write.

Our analysis itself is similar to a shape analysis, with the closest being TVLA [15]. TVLA allows developers to extend shape analysis using custom predicates that relate different objects. Our constraint specifications could be written as custom TVLA predicates, but the lower level of abstraction would result in a more complex specification and would require greater expertise from the specifier.

7 Conclusion

Relationships capture the interaction between a plugin and framework by describing how abstract object associations change as the plugin makes calls to the framework. We can then use these relationships to describe non-local constraints on the framework operations. We have shown that relationship-based constraints can describe many constraint paradigms found in real frameworks, capturing relationship structure, operation order, and object associations that may or may not derive from direct references As the specifications are written entirely by framework developers, plugin developers only need to run the analysis on their code, so that investments by a few framework developers pay dividends to many plugin developers

A modular, intra-procedural static analysis can check that the plugin code meets framework constraints. This analysis is particularly interesting because it is adjustable. While many analyses strive to only be either sound or complete, the relation analysis can be run either soundly, completely, or as a compromise of the two, thereby allowing the plugin developer to choose the variant that provides the most useful results.

References

[1] Stephanie Balzer, Thomas Gross, and Patrick Eugster. A relational model of object collaborations and its use in reasoning about relationships. In *ECOOP*, LNCS, pages 323–346. Springer, 2007.

[2] Kevin Bierhoff and Jonathan Aldrich. Modular typestate checking of aliased objects. In *OOPSLA*, pages 301–320, 2007.

[3] Gavin Bierman and Alisdair Wren. First-class relationships in an object-oriented language. In *ECOOP*, volume 3586 of *LNCS*, pages 262–286. Springer, 2005.

[4] Eric Bodden, Laurie Hendren, and Ondřej Lhoták. A staged static program analysis to improve the performance of runtime monitoring. In *ECOOP*, volume 4609 of *LNCS*, pages 525–549. Springer, 2007.

[5] Eric Bodden, Patrick Lam, and Laurie Hendren. Finding programming errors earlier by evaluating runtime monitors ahead-of-time. In *FSE*, 2008.

[6] Robert DeLine and Manuel Fähndrich. Typestates for objects. In *ECOOP*, LNCS, pages 465–490. Springer, 2004.

[7] Martin Fowler. Inversion of control containers and the dependency injection pattern. `http://www.martinfowler.com/articles/injection.html`, 2004.

[8] Richard Helm, Ian M. Holland, and Dipayan Gangopadhyay. Contracts: specifying behavioral compositions in object-oriented systems. In *OOPSLA*, pages 169–180, 1990.

[9] Daqing Hou and H. James Hoover. Using SCL to specify and check design intent in source code. *IEEE Trans. Softw. Eng.*, 32(6), 2006.

[10] Ciera Jaspan and Jonathan Aldrich. Checking semantic usage of frameworks. In *Proceedings of the 4th symposium on Library Centric Software Design*, 2007.

[11] Ralph E. Johnson. Frameworks = (components + patterns). *Commun. ACM*, 40(10), 1997.

[12] Patric Lam, Viktor Kuncak, and Martin Rinard. Generalized Typestate Checking for Data Structure Consistency. In *Verification, Model Checking, and Abstract Interpretation*, 2005.

[13] Nomair A. Naeem and Ondřej Lhoták. Typestate-like analysis of multiple interacting objects. In *OOPSLA*, pages 347–366, 2008.

[14] Mangala Gowri Nanda, Christian Grothoff, and Satish Chandra. Deriving object typestates in the presence of inter-object references. In *OOPSLA*, pages 77–96, 2005.

[15] Mooly Sagiv, Thomas Reps, and Reinhard Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Trans. Program. Lang. Syst.*, 24(3):217–298, 2002.

[16] Gang Tan, Xinming Ou, and David Walker. Enforcing resource usage protocols via scoped methods, 2003. Appeared in the 10th International Workshops on Foundations of Object-Oriented Languages.

[17] Robert J. Walker and Kevin Viggers. Implementing Protocols via Declarative Event Patterns. In *Proceedings of the 12th International symposium on Foundations of Software Engineering*, pages 159–169, 2004.

A Operations

A.1 Equivalence Join on ρ

$$\dfrac{\mathrm{dom}(\rho_l) = \mathrm{dom}(\rho_r) = \mathrm{dom}(\rho) \qquad \forall\, R \mapsto E \in \rho\,.\,E = \rho_l(R) \mathbin{\overset{=}{\sqcup}} \rho_r(R)}{\rho_l \mathbin{\overset{=}{\sqcup}} \rho_r = \rho}\,(\text{EQJOIN}-\rho)$$

A.2 Overriding Meet on ρ

$$\dfrac{\mathrm{dom}(\rho) = \mathrm{dom}(\rho_\Delta) = \mathrm{dom}(\rho') \qquad \forall\, R \mapsto E' \in \rho'\,.\,E' = \rho(R) \mathbin{\overset{\leftarrow}{\sqcap}} \rho_\Delta(R))}{\rho \mathbin{\overset{\leftarrow}{\sqcap}} \rho_\Delta = \rho'}\,(\text{OVRMEETS}-\rho)$$

A.3 Polarity operator on ρ

$$\dfrac{\mathrm{dom}(\rho) = \mathrm{dom}(\rho') \qquad \forall\, R \mapsto E \in \rho'\,.\,E = \updownarrow \rho(R)}{\updownarrow \rho = \rho'}\,(\updownarrow-\rho)$$

A.4 Join on ρ

$$\dfrac{\mathrm{dom}(\rho_l) = \mathrm{dom}(\rho_r) = \mathrm{dom}(\rho) \qquad \forall\, R \mapsto E \in \rho\,.\,E = \rho_l(R) \sqcup \rho_r(R)}{\rho_l \sqcup \rho_r = \rho}\,(\sqcup-\rho)$$

A.5 At least as precise on ρ

$$\dfrac{E_c \sqsubseteq E_a \qquad \rho_c \sqsubseteq \rho_a}{\rho_c, R \mapsto E_c \sqsubseteq \rho_a, R \mapsto E_a}\,(\sqsubseteq-\rho) \qquad\qquad \dfrac{\varnothing \sqsubseteq \rho_a}{\varnothing \sqsubseteq \rho_a, R \mapsto \mathsf{unknown}}\,(\sqsubseteq-\text{PARTIAL}-\texttt{UNKNOWN})$$

$$\dfrac{\varnothing \sqsubseteq \rho_a}{\varnothing \sqsubseteq \rho_a, R \mapsto \mathsf{bot}}\,(\sqsubseteq-\text{PARTIAL}-\texttt{BOT}) \qquad\qquad \dfrac{}{\varnothing \sqsubseteq \varnothing}\,(\sqsubseteq-\varnothing)$$

A.6 Transfer into new aliasing environment, transfer

$$\dfrac{\rho' = \{R \mapsto E \mid R \in \mathrm{dom}(\bot_{\mathcal{A}}) \ \wedge\ R \in \mathrm{dom}(\rho) \implies E = \rho(R) \ \wedge\ R \notin \mathrm{dom}(\rho) \implies E = \mathsf{unknown}\}}{\rho' = \mathsf{transfer}(\rho, \mathcal{A})}\,(\text{TRANSFER})$$

A.7 Substitution on P

$P[\sigma] = M$. Do the obvious thing.

$$\begin{array}{rcl}
(P_1 \wedge P_2)[\sigma] & = & P_1[\sigma] \wedge P_2[\sigma] \\
(P_1 \vee P_2)[\sigma] & = & P_1[\sigma] \vee P_2[\sigma] \\
(P_1 \implies P_2)[\sigma] & = & P_1[\sigma] \implies P_2[\sigma] \\
\mathit{true}[\sigma] & = & \mathit{true} \\
\mathit{false}[\sigma] & = & \mathit{false} \\
(\neg S)[\sigma] & = & \neg S[\sigma] \\
(A/y_{\mathsf{test}})[\sigma] & = & A[\sigma]/\sigma(y_{\mathsf{test}}) \\
\mathsf{rel}(\bar{y})[\sigma] & = & \mathsf{rel}(\bar{y}[\sigma]) \\
(y, \bar{y})[\sigma] & = & \sigma(y), \bar{y}[\sigma]
\end{array}$$

A.8 Lattice transformation of $\overline{\mathsf{N}}$

Notice that a list will become a pair of sets, in particular, a ρ. The sets could be conflicting, meaning that in this list, the transformation causes conflicts. We are using $\sqcup\mkern-13mu=$ to move conflicts into unknown. Alternately, we could either report this as an error or override or join. It is not clear what is best though.

$$\frac{\rho_1 = \mathit{lattice}(\mathsf{N}; \mathcal{A}; \mathcal{B}) \qquad \rho_2 = \mathit{lattice}(\overline{\mathsf{N}}; \mathcal{A}; \mathcal{B})}{\mathit{lattice}(\mathsf{N}, \overline{\mathsf{N}}; \mathcal{A}; \mathcal{B}) = \rho_1 \sqcup \rho_2}\,(\mathsf{LIST}) \qquad \frac{}{\mathit{lattice}(\mathsf{R}, \mathcal{A}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathtt{true}]}\,(\mathsf{LATTICE{-}R})$$

$$\frac{}{\mathit{lattice}(\neg\mathsf{R}, \mathcal{A}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathtt{false}]}\,(\mathsf{LATTICE{-}\neg R}) \qquad \frac{\mathcal{B}(\ell_{test}) = \mathsf{True}}{\mathit{lattice}(\mathsf{R}/\ell_{test}, \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathtt{true}]}\,(\mathsf{LATTICE{-}R{-}TEST{-}T})$$

$$\frac{\mathcal{B}(\ell_{test}) = \mathsf{False}}{\mathit{lattice}(\mathsf{R}/\ell_{test}, \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathtt{false}]}\,(\mathsf{LATTICE{-}R{-}TEST{-}F})$$

$$\frac{\mathcal{B}(\ell_{test}) = \mathsf{Unknown}}{\mathit{lattice}(\mathsf{R}/\ell_{test}, \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathsf{unknown}]}\,(\mathsf{LATTICE{-}R{-}TEST{-}U})$$

$$\frac{\mathcal{B}(\ell_{test}) = \mathsf{True}}{\mathit{lattice}(\neg\mathsf{R}/\ell_{test}, \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathtt{false}]}\,(\mathsf{LATTICE{-}\neg R{-}TEST{-}T})$$

$$\frac{\mathcal{B}(\ell_{test}) = \mathsf{False}}{\mathit{lattice}(\neg\mathsf{R}/\ell_{test}, \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathtt{true}]}\,(\mathsf{LATTICE{-}\neg R{-}TEST{-}F})$$

$$\frac{\mathcal{B}(\ell_{test}) = \mathsf{Unknown}}{\mathit{lattice}(\neg\mathsf{R}/\ell_{test}, \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[\mathsf{R} \mapsto \mathsf{unknown}]}\,(\mathsf{LATTICE{-}\neg R{-}TEST{-}U})$$

B Truth

$$\frac{}{t \preccurlyeq t}\,(\preccurlyeq{-}{=}) \qquad\qquad \frac{}{t \preccurlyeq \mathsf{Unknown}}\,(\preccurlyeq{-}\mathsf{UNKNOWN})$$

B.1 Free variables

Find the free variables and the types of a specification or a part of a specification.

$$\begin{array}{rcl}
\mathsf{FV}(\mathit{cons}) & = & \mathsf{FV}(\mathit{op}) \cup \mathsf{FV}(\mathsf{P}_{ctx}) \cup \mathsf{FV}(\mathsf{P}_{req}) \cup \mathsf{FV}(\overline{\mathsf{R}}) \\
\mathsf{FV}(\mathsf{P}_1 \wedge \mathsf{P}_2) & = & \mathsf{FV}(\mathsf{P}_1) \cup \mathsf{FV}(\mathsf{P}_2) \\
\mathsf{FV}(\mathsf{P}_1 \vee \mathsf{P}_2) & = & \mathsf{FV}(\mathsf{P}_1) \cup \mathsf{FV}(\mathsf{P}_2) \\
\mathsf{FV}(\mathsf{P}_1 \implies \mathsf{P}_2) & = & \mathsf{FV}(\mathsf{P}_1) \cup \mathsf{FV}(\mathsf{P}_2) \\
\mathsf{FV}(\mathit{true}) & = & \varnothing \\
\mathsf{FV}(\mathit{false}) & = & \varnothing \\
\mathsf{FV}(\overline{\mathsf{Q}}) & = & \bigcup \mathsf{FV}(\mathsf{Q}) \\
\mathsf{FV}(\neg\mathsf{S}) & = & \mathsf{FV}(\mathsf{S}) \\
\mathsf{FV}(\mathsf{A}/\mathsf{y}_{\mathsf{test}}) & = & \mathsf{FV}(\mathsf{A}), \mathsf{y}_{\mathsf{test}} : \mathit{boolean} \\
\mathsf{FV}(\mathsf{rel}(\bar{\mathsf{y}})) & = & \bar{\mathsf{y}} : \mathcal{R}(\mathsf{rel}) \\
 & & \\
\mathsf{FV}(\tau_{this}.\mathsf{m}(\bar{\mathsf{y}} : \bar{\tau}) : \tau_{ret}) & = & \mathsf{this} : \tau_{this}, \mathsf{ret} : \tau_{ret}, \bar{\mathsf{y}} : \bar{\tau} \\
\mathsf{FV}(\ \mathsf{new}\ \tau(\bar{\mathsf{y}} : \bar{\tau})) & = & \mathsf{this} : \tau, \bar{\mathsf{y}} : \bar{\tau}
\end{array}$$

$$\frac{}{\Gamma_y \cup \varnothing = \Gamma_y}(\cup\text{−}\varnothing) \qquad \frac{y \notin \mathrm{dom}(\Gamma_y^l) \qquad \Gamma_y^l \cup \Gamma_y^r = \Gamma_y}{\Gamma_y^l \cup y : \tau, \Gamma_y^r = y : \tau, \Gamma_y}(\cup\text{−NOTIN}) \qquad \frac{\tau^l <: \tau^r \qquad \Gamma_y^l \cup \Gamma_y^r = \Gamma_y}{y : \tau^l, \Gamma_y^l \cup y : \tau^r, \Gamma_y^r = y : \tau^l, \Gamma_y}(\cup\text{−LEFTSUB})$$

$$\frac{\tau^r <: \tau^l \qquad \Gamma_y^l \cup \Gamma_y^r = \Gamma_y}{y : \tau^l, \Gamma_y^l \cup y : \tau^r, \Gamma_y^r = y : \tau^r, \Gamma_y}(\cup\text{−RIGHT−SUB})$$

$$\frac{}{\Gamma_y - \varnothing = \Gamma_y}(\text{MINUS−}\varnothing) \qquad \frac{y \notin \mathrm{dom}(\Gamma_y^l) \qquad \Gamma_y^l \cup \Gamma_y^r = \Gamma_y}{\Gamma_y^l \cup y : \tau, \Gamma_y^r = \Gamma_y}(\text{MINUS−NOTIN})$$

$$\frac{\Gamma_y^l \cup \Gamma_y^r = \Gamma_y}{y : \tau^l, \Gamma_y^l \cup y : \tau^r, \Gamma_y^r = \Gamma_y}(\text{MINUS−IN})$$

$$\frac{\mathrm{dom}(\Gamma_y) \subseteq \mathrm{dom}(\Gamma_y') \qquad \forall y : \tau \in \Gamma_y \,.\, \Gamma_y' <: \tau}{\Gamma_y \subseteq \Gamma_y'}(\subseteq\text{−}\Gamma_Y)$$

C Aliasing Operations and Theorems

C.1 At least as precise, $\sqsubseteq_{\mathcal{A}}$

$$\frac{\mathrm{dom}(\mathcal{L}') = \mathrm{dom}(\mathcal{L}) \qquad \mathrm{dom}(\Gamma_\ell') = \mathrm{dom}(\Gamma_\ell) \qquad \forall\, \ell' : \tau' \in \Gamma_\ell' .\, \tau' <: \Gamma_\ell(\ell') \qquad \forall\, \mathtt{x}' \mapsto \bar{\ell}' \in \mathcal{L}' .\, \bar{\ell'} \subseteq \mathcal{L}(\mathtt{x}') \ \wedge\ \bar{\ell}' \neq \varnothing}{< \Gamma_\ell'; \mathcal{L}' > \sqsubseteq_{\mathcal{A}} < \Gamma_\ell; \mathcal{L} >}()(\sqsubseteq_{\mathcal{A}})$$

C.2 Abstraction function

Theorem C.1 (Abstraction of Alias Lattice from the heap)**.** *Let* $\mathtt{x} \hookrightarrow \ell : \tau$ *be a source variable* $\mathtt{x}$ *which points to a runtime location* ℓ *of type* τ*. Let* h *be a heap, represented as a list of source variables which point to locations of a particular type. Also let* H *be all the possible heaps at a particular program counter. An alias lattice* $< \Gamma_\ell, \mathcal{L} >$ *abstracts* H *at a program counter if and only if*

$$\begin{array}{l}
\forall\, h \in H \,.\, \mathrm{dom}(h) = \mathrm{dom}(\mathcal{L}) \ \wedge \\
\quad \forall\, (\mathtt{x}_1 \hookrightarrow \ell_1 : \tau_1) \in h \,.\, \forall\, (\mathtt{x}_2 \hookrightarrow \ell_2 : \tau_2) \in h \,. \\
\qquad \mathtt{x}_1 \neq \mathtt{x}_2 \ \wedge\ \ell_1 = \ell_2 \implies \\
\qquad\quad \ell' \in \mathcal{L}(\mathtt{x}_1) \ \wedge\ \ell' \in \mathcal{L}(\mathtt{x}_2) \ \wedge\ \tau_1 <: \Gamma_\ell(\ell') \wedge \\
\qquad \mathtt{x}_1 \neq \mathtt{x}_2 \ \wedge\ \ell_1 \neq \ell_2 \implies \\
\qquad\quad \ell_1' \in \mathcal{L}(\mathtt{x}_1) \ \wedge\ \ell_2' \in \mathcal{L}(\mathtt{x}_2) \ \wedge\ \ell_1' \neq \ell_2' \ \wedge\ \tau_1 <: \Gamma_\ell(\ell_1') \ \wedge\ \tau_2 <: \Gamma_\ell(\ell_2')
\end{array}$$

C.3 At least as precise, $\sqsubseteq_{\mathcal{B}}$

$$\frac{\mathrm{dom}(\mathcal{B}^c) = \mathrm{dom}(\mathcal{B}^a) \qquad \forall\, \ell : t \in \mathcal{B}^c .\, t \preccurlyeq \mathcal{B}^a(\ell)}{\mathcal{B}^c \sqsubseteq_{\mathcal{B}} \mathcal{B}^a}()(\sqsubseteq_{\mathcal{A}})$$

D Consistency

Theorem D.1. Consistency

$$\begin{array}{ll} \mathit{forall\ deriv.} & \\ & \mathcal{A} \vdash \rho \ \mathsf{consistent} \\ & \rho \ \mathsf{final} \\ & \mathit{mathitf}_{\mathit{alias}}(\mathcal{A}, \mathit{instr}) = \mathcal{A}' \\ & f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathit{instr}) = \rho' \\ \mathit{exists\ deriv.} & \\ & \mathcal{A}' \vdash \rho' \ \mathsf{consistent} \\ & \rho' \ \mathsf{final} \end{array}$$

Proof:

$\forall \mathit{cons}_i \in \mathcal{C} \,.\, \mathcal{A}'; \mathcal{B}; \rho; \mathit{cons}_i \vdash \mathit{instr} \hookrightarrow \rho_i^{\Delta}$	By inversion on $f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathit{instr}) = \rho'$
$\rho^{\Delta} = \sqcup\{\rho_i^{\Delta}\}$	By inversion on $f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathit{instr}) = \rho'$
$\rho' = \mathsf{transfer}(\rho, \mathcal{A}') \boxed{\leftarrow} \rho^{\Delta}$	By inversion on $f_{\mathcal{C};\mathcal{A};\mathcal{B}}(\rho, \mathit{instr}) = \rho'$
$\forall \ \mathit{cons}_i \in \mathcal{C} \,.\, \mathcal{A}'; \vdash \rho_i^{\Delta}$ consistent	By lemma consistency of single constraint
$\mathcal{A}' \vdash \sqcup\rho^{\Delta}$ consistent	By lemma $\sqcup$ preserves consistency
$\mathcal{A}' \vdash \mathsf{transfer}(\rho, \mathcal{A}')$ consistent	By lemma transfer implies consistency
$\mathcal{A}' \vdash \rho'$ consistent	By lemma $\boxed{\leftarrow}$ preserves consistency
ρ' final	By lemma $\boxed{\leftarrow}$ makes final

□

Theorem D.2. Consistency of a Single Constraint

$$\begin{array}{ll} \mathit{forall\ deriv.} & \\ & \mathcal{A} \vdash \rho \text{ consistent} \\ & \mathit{mathitf}_{\mathit{alias}}(\mathcal{A}, \mathit{instr}) = \mathcal{A}' \\ & \mathcal{A}'; \mathcal{B}; \rho; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho^{\Delta} \\ \mathit{exists\ deriv.} & \\ & \mathcal{A}' \vdash \rho^{\Delta} \text{ consistent} \end{array}$$

Proof:
By case analysis on $\mathcal{A}; \rho; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho^{\Delta}$

Case: $$\frac{\begin{array}{c} \mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathit{ctx}} \Rightarrow \mathsf{P}_{\mathit{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}'; \mathsf{FV}(\mathit{cons}) \vdash \mathit{instr} : \mathsf{op} \mapsto (\Sigma^{t}, \Sigma^{u}) \\ \Sigma^{t} \neq \varnothing \ \vee \ \Sigma^{u} \neq \varnothing \qquad \mathcal{P}^{t} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{t} \ \wedge \ \mathcal{A}'; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\ \mathcal{P}^{u} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{u} \ \wedge \ \mathcal{A}'; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\ |\mathcal{P}^{t}| = |\Sigma^{t}| \qquad |\mathcal{P}^{u}| = |\Sigma^{u}| \qquad \mathcal{P}^{\Delta} = \mathcal{P}^{t} \cup \mathcal{P}^{u} \end{array}}{\mathcal{A}'; \mathcal{B}; \rho; \mathit{cons} \vdash \mathit{instr} \hookrightarrow (\sqcup \mathcal{P}^{\Delta})} \text{(MATCH)}$$

$\forall\, i$.

$\mathit{dom}(\mathsf{FV}(\mathsf{op})) = \mathit{dom}(\sigma_i)$	By lemma Instruction Binding Consistent
$\mathit{rng}(\sigma_i) \subseteq \mathit{dom}(\Gamma_{\ell})$	By lemma Instruction Binding Consistent
$\forall\, \mathsf{y} : \tau \in \mathsf{FV}(\mathsf{op})\ .\ \Gamma_{\ell}(\sigma_i(\mathsf{y})) <: \tau$	By lemma Instruction Binding Consistent
$\mathcal{A} \vdash\hookrightarrow \rho_i^{\Delta}$ consistent	By lemma partial binding consistent

$\forall \rho_i^{\Delta} \in \mathcal{P}^{\Delta}\ .\ \mathcal{A} \vdash \rho_i^{\Delta}$ consistent	By quantification above
$\mathcal{A} \vdash\ \sqcup \mathcal{P}^{\Delta}$ consistent	By lemma $\sqcup$ preserves consistency

Case: $$\frac{\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathit{ctx}} \Rightarrow \mathsf{P}_{\mathit{req}} \Downarrow \overline{\mathsf{R}} \qquad \mathcal{A} \not\vdash \mathit{instr} : \mathsf{op} \mapsto \Sigma}{\mathcal{A}; \rho; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \bot_{\mathcal{A}}} \text{(NOT−MATCH)}$$

$\mathcal{R}; \mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent	By definition of $\bot_{\mathcal{A}}$

□

Theorem D.3. Consistency of Partial Binding

$$\begin{array}{l} \mathit{forall\ deriv.} \\ \qquad \mathsf{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\ \qquad \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta} \\ \mathit{exists\ deriv.} \\ \qquad \mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent} \end{array}$$

Proof:

By case analysis on $\mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}$

Case:
$$\frac{\begin{array}{c} \mathsf{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\ \Gamma_{\mathsf{y}} = \mathsf{FV}(\mathsf{op}) \cup \mathsf{FV}(\mathsf{P}_{\mathsf{ctx}}) \cup \mathsf{FV}(\overline{\mathsf{Q}}) \qquad \mathsf{allValidSubs}(\mathcal{A};\sigma_{\mathsf{op}};\Gamma_{\mathsf{y}}) = (\Sigma^{t},\Sigma^{u}) \\ \Sigma^{t} \neq \varnothing \ \vee\ \Sigma^{u} \neq \varnothing \qquad \mathcal{P}^{t} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{t} \ \wedge\ \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{full}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\ \mathcal{P}^{u} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{u} \ \wedge\ \mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{full}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \qquad \mathcal{P}^{\Delta} = \mathcal{P}^{t} \cup \mathcal{P}^{u} \end{array}}{\mathcal{A};\mathcal{B};\rho;\sigma_{\mathsf{op}} \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow (\boxminus\, \mathcal{P}^{\Delta})}\ (\textsc{Bound})$$

$\forall\ \sigma \in \Sigma^{t}\,.\,\mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \Gamma_{\mathsf{y}}$ By Lemma validSubs sound and complete

$\forall\ \sigma \in \Sigma^{u}\,.\,\mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \Gamma_{\mathsf{y}}$ By Lemma validSubs sound and complete

$\forall\ \rho^{\Delta} \in \mathcal{P}^{t}\,.$

$\mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{full}} \mathsf{cons} \hookrightarrow \rho^{\Delta}$ where $\sigma \in \Sigma^{t}$ By construction of $\mathcal{P}^{t}$

$\mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \Gamma_{\mathsf{y}}$ By $\sigma \in \Sigma^{t}$

$\mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\bar{\mathsf{Q}})$ By $\mathsf{FV}(\bar{\mathsf{Q}}) \subseteq \Gamma_{\mathsf{y}}$

$\mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}$ By Lemma Full Binding Consistent

$\forall\ \rho^{\Delta} \in \mathcal{P}^{t}\,.\,\mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}$ By quantification

$\forall\ \rho^{\Delta} \in \mathcal{P}^{u}\,.$

$\rho^{\Delta} =\updownarrow \rho^{\Delta'}$ where $\mathcal{A};\mathcal{B};\rho;\sigma \vdash_{\mathsf{full}} \mathsf{cons} \hookrightarrow \rho^{\Delta'} \ \wedge\ \sigma \in \Sigma^{u}$ By construction of $\mathcal{P}^{u}$

$\mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \Gamma_{\mathsf{y}}$ By $\sigma \in \Sigma^{u}$

$\mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\bar{\mathsf{Q}})$ By $\mathsf{FV}(\bar{\mathsf{Q}}) \subseteq \Gamma_{\mathsf{y}}$

$\mathcal{A} \vdash \rho^{\Delta'}\ \mathsf{consistent}$ By Lemma Full Binding Consistent

$\mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}$ By Lemma $\updownarrow$ consistent

$\forall\ \rho^{\Delta} \in \mathcal{P}^{u}\,.\,\mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}$ By quantification

$\forall\ \rho^{\Delta} \in \mathcal{P}^{\Delta}\,.\,\mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}$ By $\mathcal{P}^{\Delta} = \mathcal{P}^{t} \cup \mathcal{P}^{u}$

$\mathcal{A} \vdash (\boxminus\, \mathcal{P}^{\Delta})\ \mathsf{consistent}$ By Lemma $\boxminus$ consistent

Case: $$\dfrac{\begin{array}{c}\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\ \Gamma_{\mathsf{y}} = \mathsf{FV}(\mathsf{op}) \cup \mathsf{FV}(\mathsf{P}_{\mathsf{ctx}}) \cup \mathsf{FV}(\overline{\mathsf{Q}}) \qquad \mathsf{allValidSubs}(\mathcal{A}; \sigma_{\mathsf{op}}; \Gamma_{\mathsf{y}}) = (\varnothing, \varnothing)\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma_{\mathsf{op}} \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}}}\,(\text{CANT}-\text{BIND})$$

$\mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent — By definition of $\bot_{\mathcal{A}}$

□

Theorem D.4. Consistency of Full Binding

$$\begin{array}{l}
\mathit{forall\ deriv.} \\
\qquad \mathit{cons} = \mathit{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\
\qquad \mathcal{A} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\bar{\mathsf{Q}}) \\
\qquad \mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta} \\
\mathit{exists\ deriv.} \\
\qquad \mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}
\end{array}$$

Proof:

By case analysis on all variants of $\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}$

Case: $$\dfrac{\begin{array}{c}\mathit{cons} = \mathit{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{True} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathit{cons})) \\ \exists\ \sigma' \in \Sigma^{t}\ .\ \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{req}}[\sigma']\ \mathsf{True}\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])}\ (\text{FULL−T−COMPROMISE})$$

$\mathcal{A} \vdash \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])\ \mathsf{consistent}$ By Lemma Lattice with substitution is consistent

$$\dfrac{\mathit{cons} = \mathit{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{False}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}}}\ (\text{FULL−F−COMPROMISE})$$

$\mathcal{A} \vdash \bot_{\mathcal{A}}\ \mathsf{consistent}$ By definition of $\bot_{\mathcal{A}}$

$$\dfrac{\begin{array}{c}\mathit{cons} = \mathit{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\ \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{Unknown} \qquad \rho^{\Delta} = \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \updownarrow \rho^{\Delta}}\ (\text{FULL−U−COMPROMISE})$$

$\mathcal{A} \vdash \rho^{\Delta}\ \mathsf{consistent}$ By Lemma Lattice with substitution is consistent
$\mathcal{A} \vdash \updownarrow \rho^{\Delta}\ \mathsf{consistent}$ By Lemma $\updownarrow$ preserves consistency

$$\dfrac{\begin{array}{c}\mathit{cons} = \mathit{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{True} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathit{cons})) \\ \exists\ \sigma' \in \Sigma^{t}\ .\ \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{req}}[\sigma']\ \mathsf{True}\end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])}\ (\text{FULL−T−SOUND})$$

$\mathcal{A} \vdash \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])\ \mathsf{consistent}$ By Lemma Lattice with substitution is consistent

$$\frac{\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma] \text{ False}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}}}\text{(FULL–F–SOUND)}$$

$\mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent — By definition of $\bot_{\mathcal{A}}$

$$\frac{\begin{array}{c} \mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma] \text{ Unknown} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathit{cons})) \\ \exists\, \sigma' \in \Sigma^{t} \,.\, \rho\mathcal{B}; \vdash \mathsf{P}_{\mathsf{req}}[\sigma'] \text{ True} \qquad \rho^{\Delta} = \mathit{lattice}(\bar{\mathsf{Q}}[\sigma]) \end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow\updownarrow \rho^{\Delta}}\text{(FULL–U–SOUND)}$$

$\mathcal{A} \vdash \rho^{\Delta}$ consistent — By Lemma Lattice with substitution is consistent

$\mathcal{A} \vdash\updownarrow \rho^{\Delta}$ consistent — By Lemma $\updownarrow$ preserves consistency

$$\frac{\begin{array}{c} \mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma] \text{ True} \\ (\Sigma^{t}, \Sigma^{u}) = \mathsf{allValidSubs}(\mathcal{A}; \sigma; \mathsf{FV}(\mathit{cons})) \\ \exists\, \sigma' \in \Sigma^{t} \cup \Sigma^{u} \,.\, \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{req}}[\sigma'] \text{ True} \vee \rho \vdash \mathsf{P}_{\mathsf{req}}[\sigma'] \text{ Unknown} \end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])}\text{(FULL–T–COMPLETE)}$$

$\mathcal{A} \vdash \mathit{lattice}(\bar{\mathsf{Q}}[\sigma])$ consistent — By Lemma Lattice with substitution is consistent

$$\frac{\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma] \text{ False}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}}}\text{(FULL–F–COMPLETE)}$$

$\mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent — By definition of $\bot_{\mathcal{A}}$

$$\frac{\begin{array}{c} \mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\ \mathcal{B}; \rho \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma] \text{ Unknown} \qquad \rho^{\Delta} = \mathit{lattice}(\bar{\mathsf{Q}}[\sigma]) \end{array}}{\mathcal{A}; \mathcal{B}; \rho; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow\updownarrow \rho^{\Delta}}\text{(FULL–U–COMPLETE)}$$

$\mathcal{A} \vdash \rho^{\Delta}$ consistent — By Lemma Lattice with substitution is consistent

$\mathcal{A} \vdash\updownarrow \rho^{\Delta}$ consistent — By Lemma $\updownarrow$ preserves consistency

□

E Completeness

Theorem E.1. Completeness of Relations Analysis

$\mathit{forall\ der.}$

$$f_{\mathsf{alias}}(\mathcal{A}^{\mathsf{abs}}, \mathit{instr}) = \mathcal{A}^{\mathsf{abs}'}$$
$$f_{\mathsf{alias}}(\mathcal{A}^{\mathsf{conc}}, \mathit{instr}) = \mathcal{A}^{\mathsf{conc}'}$$
$$\rho^{\mathsf{abs}}\ \textsf{final}$$
$$\rho^{\mathsf{conc}}\ \textsf{final}$$
$$\mathcal{B}^{\mathsf{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathsf{abs}}$$
$$\mathcal{A}^{\mathsf{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathsf{abs}}$$
$$\mathcal{A}^{\mathsf{abs}} \vdash \rho^{\mathsf{abs}}\ \textsf{consistent}$$
$$\mathcal{A}^{\mathsf{conc}} \vdash \rho^{\mathsf{conc}}\ \textsf{consistent}$$
$$\rho^{\mathsf{conc}} \sqsubseteq \rho^{\mathsf{abs}}$$
$$f_{\mathcal{C};\mathcal{A}^{\mathsf{conc}};\mathcal{B}^{\mathsf{conc}}}(\rho^{\mathsf{conc}}, \mathit{instr}) = \rho^{\mathsf{conc}'}$$

$\mathit{exists\ der.}$

$$f_{\mathcal{C};\mathcal{A}^{\mathsf{abs}};\mathcal{B}^{\mathsf{abs}}}(\rho^{\mathsf{abs}}, \mathit{instr}) = \rho^{\mathsf{abs}'}$$
$$\rho^{\mathsf{conc}'} \sqsubseteq \rho^{\mathsf{abs}'}$$

Proof: [Completeness of Relation Analysis]

$\rho^{\mathsf{conc}'} = \mathsf{transfer}(\rho^{\mathsf{conc}}, \mathcal{A}^{\mathsf{conc}'}) \boxed{\leftarrow} \rho^{\mathsf{conc}\Delta}$	By inversion on $f_{\mathcal{C};\mathcal{A}^{\mathsf{conc}};\mathcal{B}^{\mathsf{conc}}}(\rho^{\mathsf{conc}}, \mathit{instr}) = \rho^{\mathsf{conc}'}$
$\forall\, \mathit{cons}_i \in \mathcal{C}\ .\ \mathcal{A}^{\mathsf{conc}}; \mathcal{B}^{\mathsf{conc}}\rho^{\mathsf{conc}}; \mathit{cons}_i \vdash \mathit{instr} \hookrightarrow \rho_i^{\mathsf{conc}\Delta}$	
	By inversion on $f_{\mathcal{C};\mathcal{A}^{\mathsf{conc}};\mathcal{B}^{\mathsf{conc}}}(\rho^{\mathsf{conc}}, \mathit{instr}) = \rho^{\mathsf{conc}'}$
$\rho^{\mathsf{conc}\Delta} = \sqcup\{\rho_i^{\mathsf{conc}\Delta}\}$	By inversion on $f_{\mathcal{C};\mathcal{A}^{\mathsf{conc}};\mathcal{B}^{\mathsf{conc}}}(\rho^{\mathsf{conc}}, \mathit{instr}) = \rho^{\mathsf{conc}'}$
$\forall\, \mathit{cons}_i \in \mathcal{C}\ .$	
$\quad \rho_i^{\mathsf{conc}\Delta} \sqsubseteq \rho_i^{\mathsf{abs}\Delta}$	By Lemma Soundness of Single Constraint
$\quad \mathcal{A}^{\mathsf{abs}'}; \rho^{\mathsf{abs}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho_i^{\mathsf{abs}\Delta}$	By Lemma Soundness of Single Constraint
$\quad \rho_i^{\mathsf{conc}\Delta} \trianglelefteq \rho_i^{\mathsf{abs}\Delta}$	By Lemma Soundness of Single Constraint
$\quad \mathcal{A}^{\mathsf{abs}'} \vdash \rho_i^{\mathsf{abs}\Delta}\ \textsf{consistent}$	By Lemma Consistency of Single Constraint
$\exists\, \bar{\mathsf{R}}\ .\ \forall\, i\ .\ \mathsf{dom}(\rho_i^{\mathsf{abs}\Delta}) = \bar{\mathsf{R}}$	By Lemma consistency means same domain
Let $\rho^{\mathsf{abs}\Delta} = \sqcup\{\rho_i^{\mathsf{abs}\Delta}\}$	By join rule applied many times
$\rho^{\mathsf{conc}\Delta} \trianglelefteq \rho^{\mathsf{abs}\Delta}$	By Lemma $\sqcup$ preserves $\trianglelefteq$
$\rho^{\mathsf{conc}\Delta} \sqsubseteq \rho^{\mathsf{abs}\Delta}$	By Lemma $\sqcup$ preserves $\sqsubseteq$
$\mathcal{A}^{\mathsf{abs}'} \vdash \rho^{\mathsf{abs}\Delta}\ \textsf{consistent}$	By Lemma same domains mean consistency
Let $\rho^{\mathsf{abs}''} = \mathsf{transfer}(\rho^{\mathsf{abs}}, \mathcal{A}^{\mathsf{abs}'})$	
$\mathcal{A}^{\mathsf{abs}'} \vdash \rho^{\mathsf{abs}''}\ \textsf{consistent}$	By Lemma transfer implies consistency
$\mathsf{dom}(\rho^{\mathsf{abs}''}) = \mathsf{dom}(\rho^{\mathsf{abs}\Delta})$	By Lemma consistency means same domain
Let $\rho^{\mathsf{abs}'} = \rho^{\mathsf{abs}''} \boxed{\leftarrow} \rho^{\mathsf{abs}\Delta}$	By rule overmeets
$\rho^{\mathsf{conc}'} \sqsubseteq \rho^{\mathsf{abs}'}$	By Lemma $\boxed{\leftarrow}$ preserves $\sqsubseteq$
$f_{\mathcal{C};\mathcal{A}^{\mathsf{abs}};\mathcal{B}^{\mathsf{abs}}}(\rho^{\mathsf{abs}}, \mathit{instr}) = \rho^{\mathsf{abs}'}$	By rule $\mathit{flow} - \mathit{cons}$

□

Theorem E.2. Completeness of Single Constraint

$$\begin{array}{l}
\mathit{forall\ deriv.} \\
\qquad \mathcal{A}^{\mathit{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathit{abs}} \\
\qquad \mathcal{B}^{\mathit{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathit{abs}} \\
\qquad \rho^{\mathit{conc}} \sqsubseteq \rho^{\mathit{abs}} \\
\qquad \mathcal{A}^{\mathit{abs}} \vdash \rho^{\mathit{abs}} \ \mathsf{consistent} \\
\qquad \mathcal{A}^{\mathit{conc}} \vdash \rho^{\mathit{conc}} \ \mathsf{consistent} \\
\qquad \rho^{\mathit{conc}} \ \mathsf{final} \\
\qquad \mathcal{A}^{\mathit{conc}}; \rho^{\mathit{conc}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho^{\mathit{conc}\Delta} \\
\mathit{exists\ deriv.} \\
\qquad \mathcal{A}^{\mathit{abs}}; \rho^{\mathit{abs}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho^{\mathit{abs}\Delta} \\
\qquad \rho^{\mathit{conc}\Delta} \sqsubseteq \rho^{\mathit{abs}\Delta} \\
\qquad \rho^{\mathit{conc}\Delta} \trianglelefteq \rho^{\mathit{abs}\Delta}
\end{array}$$

Proof:
By case analysis on $\mathcal{A}^{\mathit{conc}}; \rho^{\mathit{conc}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho^{\mathit{conc}\Delta}$

Case: $$\dfrac{\begin{array}{c}\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}^{\mathit{conc}}; \mathsf{FV}(\mathit{cons}) \vdash \mathit{instr} : \mathsf{op} \Mapsto (\Sigma^t_c, \Sigma^u_c) \\ \Sigma^t_c \neq \varnothing \ \lor \ \Sigma^u_c \neq \varnothing \qquad \mathcal{P}^t_c = \{\rho^{\Delta} \mid \sigma \in \Sigma^t_c \ \land \ \mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\ \mathcal{P}^u_c = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^u_c \ \land \ \mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\ |\mathcal{P}^t_c| = |\Sigma^t_c| \qquad |\mathcal{P}^u_c| = |\Sigma^u_c| \qquad \mathcal{P}^{\Delta}_c = \mathcal{P}^t_c \cup \mathcal{P}^u_c \end{array}}{\mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow (\bigsqcup \mathcal{P}^{\Delta}_c)} \ (\mathsf{MATCH})$$

Let $\rho^{\Delta}_a = \hookrightarrow (\bigsqcup \mathcal{P}^{\Delta}_a)$
$\mathcal{A}^{\mathit{conc}}; \mathsf{FV}(\mathit{cons}) \vdash \mathit{instr} : \mathsf{op} \Mapsto (\Sigma^t_a, \Sigma^u_a)$ By Lemma Instruction Binding Complete
$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By Lemma Instruction Binding Complete
$\Sigma^u_c \subseteq \Sigma^u_a$ By Lemma Instruction Binding Complete
$\Sigma^t_c \supseteq \Sigma^t_a$ By Lemma Instruction Binding Complete
$\Sigma^t_a \neq \varnothing \lor \Sigma^u_a \neq \varnothing$ By $\Sigma^t_c \neq \varnothing \ \lor \ \Sigma^u_c \neq \varnothing$ and inversion on $\subseteq$
Let $\mathcal{P}^t_a = \{\rho^{\Delta} \mid \sigma \in \Sigma^t_a \ \land \ \mathcal{A}^{\mathit{abs}}; \mathcal{B}^{\mathit{abs}}; \rho^{\mathit{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\}$
Let $\mathcal{P}^u_a = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^u_a \ \land \ \mathcal{A}^{\mathit{abs}}; \mathcal{B}^{\mathit{abs}}; \rho^{\mathit{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\}$
$\forall\ \rho^{t\Delta}_c \in \mathcal{P}^t_c$.

$\exists$ distinct $\sigma^t \in \Sigma^t_c$. $\mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{t\Delta}_c$
By construction of $\mathcal{P}^t_c$ and $|\Sigma^t_c| = |\mathcal{P}^t_c|$
$\sigma^t \in \Sigma^t_a \ \lor \ \sigma^t \in \Sigma^u_a$ By $\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By case analysis on the location of σ^t

Case: $\sigma^t \in \Sigma^t_a$

$\mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_a^{t\Delta}$ By Lemma Partial Binding complete

$\rho_c^{t\Delta} \sqsubseteq \rho_a^{t\Delta}$ By Lemma Partial Binding Sound

$\rho_c^{t\Delta} \trianglelefteq \rho_a^{t\Delta}$ By Lemma Partial Binding Sound

$\rho_a^{t\Delta}$ distinct $\in \mathcal{P}_a^t$ By construction of $\mathcal{P}_a^t$

Case: $\sigma^u \in \Sigma_a^u$

$\mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_a^{u\Delta}$ By Lemma Partial Binding complete

$\rho_c^{t\Delta} \sqsubseteq \rho_a^{u\Delta}$ By Lemma Partial Binding Sound

$\rho_c^{t\Delta} \trianglelefteq \rho_a^{u\Delta}$ By Lemma Partial Binding Sound

$\rho_c^{t\Delta} \sqsubseteq\updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ on abs preserves $\sqsubseteq$

$\rho_c^{t\Delta} \trianglelefteq \updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ on abs preserves $\trianglelefteq$

$\rho_a^{u\Delta}$ distinct $\in \mathcal{P}_a^u$ By construction of $\mathcal{P}_a^u$

$\forall\ \rho_c^{u\Delta} \in \mathcal{P}_c^u$.

$\exists$ distinct $\sigma^u \in \Sigma_c^u$. $\mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_c^{u\Delta'}$

By construction of $\mathcal{P}_c^u$ and $|\Sigma_c^u| = |\mathcal{P}_c^u|$

$\rho_c^{u\Delta} =\updownarrow \rho_c^{u\Delta'}$ By construction of $\mathcal{P}_c^u$

$\sigma^u \in \Sigma_a^u$ By $\Sigma_c^u \subseteq \Sigma_a^u$

$\mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_a^{u\Delta'}$ By Lemma Partial Binding complete

$\rho_c^{u\Delta'} \sqsubseteq \rho_a^{u\Delta'}$ By Lemma Partial Binding Sound

$\rho_c^{u\Delta'} \trianglelefteq \rho_a^{u\Delta'}$ By Lemma Partial Binding Sound

$\rho_c^{u\Delta} \sqsubseteq\updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ preserves $\sqsubseteq$

$\rho_c^{u\Delta} \trianglelefteq \updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ preserves $\trianglelefteq$

$\rho_a^{u\Delta}$ distinct $\in \mathcal{P}_a^u$ By construction of $\mathcal{P}_a^u$

$\forall\ \rho_c^{t\Delta} \in \mathcal{P}_c^t$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{t\Delta} \sqsubseteq \rho_a^{\Delta}$ By quantification above

$\forall\ \rho_c^{t\Delta} \in \mathcal{P}_c^t$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{t\Delta} \trianglelefteq \rho_a^{\Delta}$ By quantification above

$\forall\ \rho_c^{u\Delta} \in \mathcal{P}_c^u$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{u\Delta} \sqsubseteq \rho_a^{\Delta}$ By quantification above

$\forall\ \rho_c^{u\Delta} \in \mathcal{P}_c^u$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{u\Delta} \trianglelefteq \rho_a^{\Delta}$ By quantification above

$\forall\ \rho_a^{t\Delta} \in \mathcal{P}_a^t$. $\mathcal{A}_{\mathrm{abs}} \vdash \rho_a^{t\Delta}$ consistent By quantification above

$\forall\ \rho_a^{u\Delta} \in \mathcal{P}_a^u$. $\mathcal{A}_{\mathrm{abs}} \vdash \rho_a^{u\Delta}$ consistent By quantification above

Let $\mathcal{P}_a = \mathcal{P}_a^t \cup \mathcal{P}_a^u$

$\exists \bar{R}$. $\forall\ \rho_a \in \mathcal{P}_a$. $\mathrm{dom}(\rho_a) = \bar{R}$ By inversion on consistency of each ρ_a

Let $\rho_a^{\Delta} = (\ \boxminus\ \mathcal{P}_a)$

$\mathcal{A}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \rho^{\mathrm{abs}\Delta}$ By rule match

$\forall\ \rho_c^{\Delta} \in \mathcal{P}_c$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{\Delta} \sqsubseteq \rho_a^{\Delta}$ By $\mathcal{P}_c = \mathsf{Rho}_c^t \cup \mathsf{Rho}_c^u$

$\forall\ \rho_c^{\Delta} \in \mathcal{P}_c$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{\Delta} \trianglelefteq \rho_a^{\Delta}$ By $\mathcal{P}_c = \mathsf{Rho}_c^t \cup \mathsf{Rho}_c^u$

$\rho_c^{\Delta} \sqsubseteq \rho_a^{\Delta}$ By $\boxminus$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

$\rho_c^{\Delta} \trianglelefteq \rho_a^{\Delta}$ By $\boxminus$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

Case: $$\frac{\mathit{cons} = \mathit{op} : \mathsf{P}_{\mathit{ctx}} \Rightarrow \mathsf{P}_{\mathit{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}^{\mathrm{conc}}; \mathsf{FV}(\mathit{cons}) \vdash \mathit{instr} : \mathit{op} \mapsto (\varnothing, \varnothing)}{\mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \bot_{\mathcal{A}^{\mathrm{conc}}}}\ (\text{NO-MATCH})$$

$\mathsf{allValidSubs}(\mathcal{A}^{abs}; \sigma_{op}; \Gamma_y) \mapsto (\Sigma^t_a, \Sigma^u_a)$ By Lemma Instruction Binding Complete

$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By Lemma Instruction Binding Complete

$\Sigma^u_c \subseteq \Sigma^u_a$ By Lemma Instruction Binding Complete

$\Sigma^t_c \supseteq \Sigma^t_a$ By Lemma Instruction Binding Complete

$\Sigma^t_a = \varnothing$ By $\Sigma^t_c \supseteq \Sigma^t_a$

By case analysis on the property $\Sigma^u_a = \varnothing$

Case: $\Sigma^u_a = \varnothing$

$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; cons \vdash instr \hookrightarrow \bot_{\mathcal{A}^{abs}}$ By rule $no - match$

$\bot^{conc}_{\mathcal{A}} \sqsubseteq \bot^{abs}_{\mathcal{A}}$ By definition of $\bot_{\mathcal{A}}$

$\bot^{conc}_{\mathcal{A}} \trianglelefteq \bot^{abs}_{\mathcal{A}}$ By definition of $\bot_{\mathcal{A}}$

Case: $\Sigma^u_a \neq \varnothing$

Let $\mathcal{P}^t_a = \{\rho^\Delta \mid \sigma \in \Sigma^t_a \ \wedge\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{part}} cons \hookrightarrow \rho^\Delta\}$ $\mathcal{P}^t_a = \varnothing$ By $\Sigma^t_a = \varnothing$

Let $\mathcal{P}^u_c = \{\updownarrow \rho^{\Delta'} \mid \sigma \in \Sigma^u_c \ \wedge\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{part}} cons \hookrightarrow \rho^{\Delta'}\}$

$\forall\, R \mapsto E \in \bot_{\mathcal{A}^{conc}} \,.\, E = \mathtt{bot}$ By definition of $\bot_{\mathcal{A}}$

$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}^{conc}}$ consistent By definition of $\bot_{\mathcal{A}}$

$\forall\, \rho^\Delta \in \mathcal{P}^u_c \,.$

$\rho^\Delta = \updownarrow \rho^{\Delta'}$ where $\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{part}} cons \hookrightarrow \rho^{\Delta'}$ By construction of $\mathcal{P}^u_c$

$\mathcal{A}^{abs} \vdash \rho^{\Delta'}$ consistent By lemma partial binding consistent

$\mathcal{A}^{abs} \vdash \rho^\Delta$ consistent By lemma $\updownarrow$ consistent

$dom(\bot_{\mathcal{A}^{conc}}) \subseteq dom(\rho^\Delta)$ By Lemma consistency and $\sqsubseteq_{\mathcal{A}}$ implies domains subset

$\forall\, R \mapsto E \in\ \rho^\Delta \,.\, E = \mathtt{bot} \ \vee\ E = \mathtt{unknown}$ By $\updownarrow$ creates polarity

$\forall\, R \mapsto E \in \bot_{\mathcal{A}^{conc}} \,.\, E \sqsubseteq \rho^\Delta(R)$ By rule $\sqsubseteq -\mathtt{bot}$

$\bot_{\mathcal{A}^{conc}} \sqsubseteq \rho^\Delta$ By rule $\sqsubseteq -\rho$

$\forall\, R \mapsto E \in \bot_{\mathcal{A}^{conc}} \,.\, E \trianglelefteq \rho^\Delta(R)$ By rule $\trianglelefteq - \mathtt{bot}$ and $\trianglelefteq - \mathtt{unknown}$

$\bot_{\mathcal{A}^{conc}} \trianglelefteq \rho^\Delta$ By rule $\trianglelefteq - \rho$

$\mathcal{A}^{abs}; \rho^{abs}; cons \vdash instr \hookrightarrow \rho^{abs\Delta}$ By rule $match$

$\forall\, \rho^\Delta \in \mathcal{P}^u_c \,.\, \bot_{\mathcal{A}^{conc}} \sqsubseteq \rho^\Delta$ By quantification

$\forall\, \rho^\Delta \in \mathcal{P}^u_c \,.\, \bot_{\mathcal{A}^{conc}} \trianglelefteq \rho^\Delta$ By quantification

$\bot_{\mathcal{A}^{conc}} \sqsubseteq (\mathrel{\sqcup\mkern-13mu=}\, \mathcal{P}^u_c)$ By lemma $\mathrel{\sqcup\mkern-13mu=}$ preserves $\sqsubseteq$

$\bot_{\mathcal{A}^{conc}} \trianglelefteq (\mathrel{\sqcup\mkern-13mu=}\, \mathcal{P}^u_c)$ By lemma $\mathrel{\sqcup\mkern-13mu=}$ preserves $\trianglelefteq$

□

Theorem E.3. Completeness of Constraint with Partial Substitution

$$
\begin{array}{ll}
\mathit{forall\ deriv.} & \\
& \mathcal{A}^{\mathrm{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathrm{abs}} \\
& \rho^{\mathrm{conc}} \sqsubseteq \rho^{\mathrm{abs}} \\
& \rho^{\mathrm{abs}}\ \mathsf{final} \\
& \rho^{\mathrm{conc}}\ \mathsf{final} \\
& \mathcal{A}^{\mathrm{abs}} \vdash \rho^{\mathrm{abs}}\ \mathsf{consistent} \\
& \mathcal{A}^{\mathrm{conc}} \vdash \rho^{\mathrm{conc}}\ \mathsf{consistent} \\
& \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \longrightarrow \rho^{\mathrm{conc}\Delta} \\
& \mathcal{B}^{\mathrm{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathrm{abs}} \\
\mathit{exists\ deriv.} & \\
& \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \longrightarrow \rho^{\mathrm{abs}\Delta} \\
& \rho^{\mathrm{conc}\Delta} \sqsubseteq \rho^{\mathrm{abs}\Delta} \\
& \rho^{\mathrm{conc}\Delta} \trianglelefteq \rho^{\mathrm{abs}\Delta}
\end{array}
$$

Proof:
By case analysis on $\mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \longrightarrow \rho^{\mathrm{conc}\Delta}$

Case:
$$
\frac{\begin{array}{c}
\mathit{cons} = \mathrm{op} : \mathrm{P}_{\mathrm{ctx}} \Rightarrow \mathrm{P}_{\mathrm{req}} \Downarrow \overline{\mathrm{Q}} \\
\Gamma_{\mathrm{y}} = \mathsf{FV}(\mathrm{op}) \cup \mathsf{FV}(\mathrm{P}_{\mathrm{ctx}}) \cup \mathsf{FV}(\overline{\mathrm{Q}}) \qquad \mathsf{allValidSubs}(\mathcal{A}^{\mathrm{conc}}; \sigma_{\mathrm{op}}; \Gamma_{\mathrm{y}}) = (\Sigma^{\mathrm{t}}_{\mathrm{c}}, \Sigma^{\mathrm{u}}_{\mathrm{c}}) \\
\Sigma^{\mathrm{t}}_{\mathrm{c}} \neq \varnothing \ \vee\ \Sigma^{\mathrm{u}}_{\mathrm{c}} \neq \varnothing \qquad \mathcal{P}^{\mathrm{t}}_{\mathrm{c}} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{\mathrm{t}}_{\mathrm{c}} \ \wedge\ \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\
\mathcal{P}^{\mathrm{u}}_{\mathrm{c}} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{\mathrm{u}}_{\mathrm{c}} \ \wedge\ \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\
|\Sigma^{\mathrm{t}}_{\mathrm{c}}| = |\mathcal{P}^{\mathrm{t}}_{\mathrm{c}}| \qquad |\Sigma^{\mathrm{u}}_{\mathrm{c}}| = |\mathcal{P}^{\mathrm{u}}_{\mathrm{c}}| \qquad \mathcal{P}^{\Delta}_{\mathrm{c}} = \mathcal{P}^{\mathrm{t}}_{\mathrm{c}} \cup \mathcal{P}^{\mathrm{u}}_{\mathrm{c}}
\end{array}}{\mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma_{\mathrm{op}} \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow (\bigsqcup \mathcal{P}^{\Delta})} \text{(BOUND)}
$$

Let $\rho^{\Delta}_{\mathrm{a}} = \hookrightarrow (\bigsqcup \mathcal{P}^{\Delta}_{\mathrm{a}})$

$\mathsf{allValidSubs}(\mathcal{A}^{\mathrm{abs}}; \sigma_{\mathrm{op}}; \Gamma_{\mathrm{y}}) = (\Sigma^{\mathrm{t}}_{\mathrm{a}}, \Sigma^{\mathrm{u}}_{\mathrm{a}})$ By Lemma All Valid Subs sound and complete

$\Sigma^{\mathrm{t}}_{\mathrm{c}} \subseteq \Sigma^{\mathrm{t}}_{\mathrm{a}} \cup \Sigma^{\mathrm{u}}_{\mathrm{a}}$ By Lemma All Valid Subs sound and complete

$\Sigma^{\mathrm{u}}_{\mathrm{c}} \subseteq \Sigma^{\mathrm{u}}_{\mathrm{a}}$ By Lemma All Valid Subs sound and complete

$\Sigma^{\mathrm{t}}_{\mathrm{c}} \supseteq \Sigma^{\mathrm{t}}_{\mathrm{a}}$ By Lemma All Valid Subs sound and complete

$\Sigma^{\mathrm{t}}_{\mathrm{a}} \neq \varnothing \vee \Sigma^{\mathrm{u}}_{\mathrm{a}} \neq \varnothing$ By $\Sigma^{\mathrm{t}}_{\mathrm{c}} \neq \varnothing \ \vee\ \Sigma^{\mathrm{u}}_{\mathrm{c}} \neq \varnothing$ and inversion on $\subseteq$

Let $\mathcal{P}^{\mathrm{t}}_{\mathrm{a}} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{\mathrm{t}}_{\mathrm{a}} \ \wedge\ \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\}$

Let $\mathcal{P}^{\mathrm{u}}_{\mathrm{a}} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{\mathrm{u}}_{\mathrm{a}} \ \wedge\ \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\}$

$\forall\ \rho^{\mathrm{t}\Delta}_{\mathrm{c}} \in \mathcal{P}^{\mathrm{t}}_{\mathrm{c}}$.

$\exists$ distinct $\sigma^{\mathrm{t}} \in \Sigma^{\mathrm{t}}_{\mathrm{c}}$. $\mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\mathrm{t}\Delta}_{\mathrm{c}}$By construction of $\mathcal{P}^{\mathrm{t}}_{\mathrm{c}}$ and $|\Sigma^{\mathrm{t}}_{\mathrm{c}}| = |\mathcal{P}^{\mathrm{t}}_{\mathrm{c}}|$

$\sigma^{\mathrm{t}} \in \Sigma^{\mathrm{t}}_{\mathrm{a}} \ \vee\ \sigma^{\mathrm{t}} \in \Sigma^{\mathrm{u}}_{\mathrm{a}}$ By $\Sigma^{\mathrm{t}}_{\mathrm{c}} \subseteq \Sigma^{\mathrm{t}}_{\mathrm{a}} \cup \Sigma^{\mathrm{u}}_{\mathrm{a}}$

By case analysis on the location of σ^{t}

Case: $\sigma^{\mathrm{t}} \in \Sigma^{\mathrm{t}}_{\mathrm{a}}$

$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_a^{t\Delta}$ By Lemma Full complete
$\rho_c^{t\Delta} \sqsubseteq \rho_a^{t\Delta}$ By Lemma Full complete
$\rho_c^{t\Delta} \trianglelefteq \rho_a^{t\Delta}$ By Lemma Full complete
$\rho_a^{t\Delta}$ distinct $\in \mathcal{P}_a^t$ By construction of $\mathcal{P}_a^t$

Case: $\sigma^u \in \Sigma_a^u$
$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_a^{u\Delta}$ By Lemma Full complete
$\rho_c^{t\Delta} \sqsubseteq \rho_a^{u\Delta}$ By Lemma Full complete
$\rho_c^{t\Delta} \trianglelefteq \rho_a^{u\Delta}$ By Lemma Full complete
$\rho_c^{t\Delta} \sqsubseteq\!\updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ on abs preserves $\sqsubseteq$
$\rho_c^{t\Delta} \trianglelefteq \updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ on abs preserves $\trianglelefteq$
$\rho_a^{u\Delta}$ distinct $\in \mathcal{P}_a^u$ By construction of $\mathcal{P}_a^u$

$\forall\ \rho_c^{u\Delta} \in \mathcal{P}_c^u$.

$\exists$ distinct $\sigma^u \in \Sigma_c^u$. $\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_c^{u\Delta'}$By construction of $\mathcal{P}_c^u$ and $|\Sigma_c^u| = |\mathcal{P}_c^u|$
$\rho_c^{u\Delta} =\!\updownarrow \rho_c^{u\Delta'}$ By construction of $\mathcal{P}_c^u$
$\sigma^u \in \Sigma_a^u$ By $\Sigma_c^u \subseteq \Sigma_a^u$
$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_a^{u\Delta'}$ By Lemma Full complete
$\rho_c^{u\Delta'} \sqsubseteq \rho_a^{u\Delta'}$ By Lemma Full complete
$\rho_c^{u\Delta'} \trianglelefteq \rho_a^{u\Delta'}$ By Lemma Full complete
$\rho_c^{u\Delta} \sqsubseteq\!\updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ preserves $\sqsubseteq$
$\rho_c^{u\Delta} \trianglelefteq \updownarrow \rho_a^{u\Delta}$ By Lemma $\updownarrow$ preserves $\trianglelefteq$
$\rho_a^{u\Delta}$ distinct $\in \mathcal{P}_a^u$ By construction of $\mathcal{P}_a^u$

$\forall\ \rho_c^{t\Delta} \in \mathcal{P}_c^t$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{t\Delta} \sqsubseteq \rho_a^{\Delta}$ By quantification above
$\forall\ \rho_c^{t\Delta} \in \mathcal{P}_c^t$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{t\Delta} \trianglelefteq \rho_a^{\Delta}$ By quantification above
$\forall\ \rho_c^{u\Delta} \in \mathcal{P}_c^u$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{u\Delta} \sqsubseteq \rho_a^{\Delta}$ By quantification above
$\forall\ \rho_c^{u\Delta} \in \mathcal{P}_c^u$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{u\Delta} \trianglelefteq \rho_a^{\Delta}$ By quantification above
$\forall\ \rho_a^{t\Delta} \in \mathcal{P}_a^t$. $\mathcal{A}_{abs} \vdash \rho_a^{t\Delta}$ consistent By quantification above
$\forall\ \rho_a^{u\Delta} \in \mathcal{P}_a^u$. $\mathcal{A}_{abs} \vdash \rho_a^{u\Delta}$ consistent By quantification above
Let $\mathcal{P}_a = \mathcal{P}_a^t \cup \mathcal{P}_a^u$
$\exists \bar{R}\ .\ \forall\ \rho_a \in \mathcal{P}_a\ .\ \mathrm{dom}(\rho_a) = \bar{R}$ By inversion on consistency of each ρ_a
Let $\rho_a^{\Delta} = (\ \boxminus\, \mathcal{P}_a)$
$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs} \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^{abs\Delta}$ By rule bind
$\forall\ \rho_c^{\Delta} \in \mathcal{P}_c$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{\Delta} \sqsubseteq \rho_a^{\Delta}$ By $\mathcal{P}_c = \mathsf{Rho}_c^t \cup \mathsf{Rho}_c^u$
$\forall\ \rho_c^{\Delta} \in \mathcal{P}_c$. $\exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a$. $\rho_c^{\Delta} \trianglelefteq \rho_a^{\Delta}$ By $\mathcal{P}_c = \mathsf{Rho}_c^t \cup \mathsf{Rho}_c^u$
$\rho_c^{\Delta} \sqsubseteq \rho_a^{\Delta}$ By $\boxminus$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets
$\rho_c^{\Delta} \trianglelefteq \rho_a^{\Delta}$ By $\boxminus$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

Case:
$$\frac{\mathit{cons} = \underline{\mathsf{op}} : \mathsf{P}_{ctx} \Rightarrow \mathsf{P}_{req} \Downarrow \overline{\mathsf{Q}} \qquad \Gamma_y = \mathsf{FV}(\mathsf{op}) \cup \mathsf{FV}(\mathsf{P}_{ctx}) \cup \mathsf{FV}(\overline{\mathsf{Q}}) \qquad \mathsf{allValidSubs}(\mathcal{A}^{conc}; \sigma_{op}; \Gamma_y) = (\varnothing, \varnothing)}{\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma_{op} \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}^{conc}}}\ (\mathsf{CANT\text{-}BIND})$$

$\mathsf{allValidSubs}(\mathcal{A}^{abs}; \sigma_{op}; \Gamma_y) \mapsto (\Sigma^t_a, \Sigma^u_a)$	By Lemma All Subs Sound and complete
$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$	By Lemma All Subs Sound and complete
$\Sigma^u_c \subseteq \Sigma^u_a$	By Lemma All Subs Sound and complete
$\Sigma^t_c \supseteq \Sigma^t_a$	By Lemma All Subs Sound and complete
$\Sigma^t_a = \varnothing$	By $\Sigma^t_c \supseteq \Sigma^t_a$

By case analysis on the property $\Sigma^u_a = \varnothing$

Case: $\Sigma^u_a = \varnothing$

$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; cons \vdash instr \hookrightarrow \bot_{\mathcal{A}^{abs}}$	By rule $cant - bind$
$\bot^{conc}_{\mathcal{A}} \sqsubseteq \bot^{abs}_{\mathcal{A}}$	By definition of $\bot_{\mathcal{A}}$
$\bot^{conc}_{\mathcal{A}} \trianglelefteq \bot^{abs}_{\mathcal{A}}$	By definition of $\bot_{\mathcal{A}}$

Case: $\Sigma^u_a \neq \varnothing$

Let $\mathcal{P}^t_a = \{\rho^\Delta \mid \sigma \in \Sigma^t_a \ \wedge\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow \rho^\Delta\}\ \mathcal{P}^t_a = \varnothing$	By $\Sigma^t_a = \varnothing$
Let $\mathcal{P}^u_c = \{\updownarrow \rho^{\Delta'} \mid \sigma \in \Sigma^u_c \ \wedge\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{\Delta'}\}$	
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}^{conc}}\ .\ \mathsf{E} = \mathtt{bot}$	By definition of $\bot_{\mathcal{A}}$
$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}^{conc}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$\forall\, \rho^\Delta \in \mathcal{P}^u_c\ .$	

$\rho^\Delta = \updownarrow \rho^{\Delta'}$ where $\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{\Delta'}$	By construction of $\mathcal{P}^u_c$
$\mathcal{A}^{abs} \vdash \rho^{\Delta'}$ consistent	By lemma full consistent
$\mathcal{A}^{abs} \vdash \rho^{\Delta}$ consistent	By lemma $\updownarrow$ consistent
$dom(\bot_{\mathcal{A}^{conc}}) \subseteq dom(\rho^\Delta)$	By Lemma consistency and $\sqsubseteq_{\mathcal{A}}$ implies domains subset
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in\ \rho^\Delta\ .\ \mathsf{E} = \mathtt{bot}\ \vee\ \mathsf{E} = \mathsf{unknown}$	By $\updownarrow$ creates polarity
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}^{conc}}\ .\ \mathsf{E} \sqsubseteq \rho^\Delta(\mathsf{R})$	By rule $\sqsubseteq -\mathtt{bot}$
$\bot_{\mathcal{A}^{conc}} \sqsubseteq \rho^\Delta$	By rule $\sqsubseteq -\rho$
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}^{conc}}\ .\ \mathsf{E} \trianglelefteq \rho^\Delta(\mathsf{R})$	By rule $\trianglelefteq - \mathtt{bot}$ and $\trianglelefteq - \mathsf{unknown}$
$\bot_{\mathcal{A}^{conc}} \trianglelefteq \rho^\Delta$	By rule $\trianglelefteq - \rho$

$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs} \vdash_{\mathsf{part}} cons \hookrightarrow \rho^{abs\Delta}$	By rule $bind$
$\forall\, \rho^\Delta \in \mathcal{P}^u_c\ .\ \bot_{\mathcal{A}^{conc}} \sqsubseteq \rho^\Delta$	By quantification
$\forall\, \rho^\Delta \in \mathcal{P}^u_c\ .\ \bot_{\mathcal{A}^{conc}} \trianglelefteq \rho^\Delta$	By quantification
$\bot_{\mathcal{A}^{conc}} \sqsubseteq (\ {\sqcup\!\!\!\!\!=}\ \mathcal{P}^u_c)$	By lemma ${\sqcup\!\!\!\!\!=}$ preserves $\sqsubseteq$
$\bot_{\mathcal{A}^{conc}} \trianglelefteq (\ {\sqcup\!\!\!\!\!=}\ \mathcal{P}^u_c)$	By lemma ${\sqcup\!\!\!\!\!=}$ preserves $\trianglelefteq$

□

Theorem E.4. Completeness of Constraint with Full Substitution

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathit{forall\ deriv.}} \\
 & \mathcal{A}^{\mathit{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathit{abs}} \\
 & \mathcal{B}^{\mathit{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathit{abs}} \\
 & \rho^{\mathit{conc}} \sqsubseteq \rho^{\mathit{abs}} \\
 & \mathcal{A}^{\mathit{abs}} \vdash \rho^{\mathit{abs}}\ \mathsf{consistent} \\
 & \mathcal{A}^{\mathit{conc}} \vdash \rho^{\mathit{conc}}\ \mathsf{consistent} \\
 & \rho^{\mathit{abs}}\ \mathsf{final} \\
 & \rho^{\mathit{conc}}\ \mathsf{final} \\
 & \mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \longrightarrow \rho^{\mathit{conc}\Delta} \\
 & \mathit{dom}(\sigma) = \mathit{dom}(\mathit{FV}(\mathit{cons})) \\
\multicolumn{2}{l}{\mathit{exists\ deriv.}} \\
 & \mathcal{A}^{\mathit{abs}}; \mathcal{B}^{\mathit{abs}}; \rho^{\mathit{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \longrightarrow \rho^{\mathit{abs}\Delta} \\
 & \rho^{\mathit{conc}\Delta} \sqsubseteq \rho^{\mathit{abs}\Delta} \\
 & \rho^{\mathit{conc}\Delta} \trianglelefteq \rho^{\mathit{abs}\Delta}
\end{array}
$$

Proof:
By case analysis on $\mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \longrightarrow \rho^{\mathit{conc}\Delta}$

Case: $$\frac{\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rrightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{False}}{\mathcal{A}^{\mathit{conc}}; \mathcal{B}^{\mathit{conc}}; \rho^{\mathit{conc}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}^{\mathit{conc}}}}\ (\mathsf{FULL{-}F{-}COMPLETE})$$

$\mathcal{B}^{\mathit{abs}}; \rho^{\mathit{abs}} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ t^{a}$	By lemma truth sound
$\mathsf{False} \preccurlyeq t^{a}$	By lemma truth sound

By case analysis on the value of t^{a}

Case: $t^{a} = \mathsf{False}$

$\mathcal{A}^{\mathit{abs}}; \mathcal{B}^{\mathit{abs}}; \rho^{\mathit{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \bot_{\mathcal{A}}^{\mathit{abs}}$	By rule $\mathit{full} - \mathit{complete} - \mathsf{False}$
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}}^{\mathit{conc}}\ .\ \mathsf{E} = \mathtt{bot}$	By definition of $\bot$
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}}^{\mathit{conc}}\ .\ \mathsf{E} \sqsubseteq \bot_{\mathcal{A}}^{\mathit{abs}}(\mathsf{R})$	By rule $\sqsubseteq - \bot$
$\bot_{\mathcal{A}}^{\mathit{conc}} \sqsubseteq \bot_{\mathcal{A}}^{\mathit{abs}}$	By rule $\sqsubseteq$
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}}^{\mathit{abs}}\ .\ \mathsf{E} = \mathtt{bot}$	By definition of $\bot$
$\forall\, \mathsf{R} \mapsto \mathsf{E} \in \rho^{\mathit{conc}\Delta}\ .\ \mathsf{E} \trianglelefteq \rho^{\mathit{abs}\Delta}(\mathsf{R})$	By rule $\trianglelefteq - \mathtt{bot}$
$\bot_{\mathcal{A}}^{\mathit{conc}} \trianglelefteq \bot_{\mathcal{A}}^{\mathit{abs}}$	By rule $\trianglelefteq$

Case: $t^{a} = \mathsf{True}$

Invalid case by $\mathsf{False} \preccurlyeq t^{a}$

Case: $\mathsf{t}^{a} = \mathsf{Unknown}$

Let $\rho_a^{\Delta'} = \mathtt{lattice}(\bar{Q}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})$

Let $\rho_a^{\Delta} =\updownarrow \rho^{\Delta'}$

$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \rho_a^{\Delta}$ By rule $\mathtt{full} - \mathtt{complete} - \mathsf{Unknown}$

$\mathcal{A}^{abs} \vdash \rho_a^{\Delta'}$ consistent By lattice consistent

$\mathcal{A}^{abs} \vdash \rho_a^{\Delta}$ consistent By $\updownarrow$ consistent

$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}^{conc}}$ consistent By definition of $\bot_{\mathcal{A}}$

$\mathtt{dom}(\bot_{\mathcal{A}^{conc}}) \subseteq \mathtt{dom}(\rho_a^{\Delta})$ By consistency and $\sqsubseteq_{\mathcal{A}}$ implies domains subset

$\forall\, R \mapsto E \in \bot_{\mathcal{A}}^{conc} \,.\, E = \mathtt{bot}$ By definition of $\bot$

$\forall\, R \mapsto E \in \rho_a^{\Delta} \,.\, E = \mathtt{bot} \;\vee\; E = \mathbf{unknown}$ By $\updownarrow$ creates polarity

$\forall\, R \mapsto E \in \bot_{\mathcal{A}}^{conc} \,.\, E \sqsubseteq \rho_a^{\Delta}(R)$ By rule $\sqsubseteq -\mathtt{bot}$

$\bot_{\mathcal{A}}^{conc} \sqsubseteq \rho^{abs\Delta}$ By rule $\sqsubseteq$

$\forall\, R \mapsto E \in \rho^{conc\Delta} \,.\, E \trianglelefteq \rho^{abs\Delta}(R)$ By rule $\trianglelefteq - \mathtt{bot}$ and $\trianglelefteq - \mathbf{unknown}$

$\bot_{\mathcal{A}}^{conc} \trianglelefteq \rho^{abs\Delta}$ By rule $\trianglelefteq$

Case:
$$\frac{\begin{array}{c}\mathtt{cons} = \mathsf{op} : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \qquad \mathcal{B}^{conc}; \rho^{conc} \vdash P_{ctx}[\sigma]\ \mathsf{True} \\ (\Sigma_c^t, \Sigma_c^u) = \mathsf{allValidSubs}(\mathcal{A}^{conc}; \sigma; \mathsf{FV}(\mathtt{cons})) \\ \exists\, \sigma' \in \Sigma_c^t \cup \Sigma_c^u \,.\, \mathcal{B}^{conc}; \rho^{conc} \vdash P_{req}[\sigma']\ \mathsf{True} \vee \rho^{conc} \vdash P_{req}[\sigma']\ \mathsf{Unknown}\end{array}}{\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \mathtt{lattice}(\bar{Q}[\sigma]; \mathcal{A}^{conc}; \mathcal{B}^{conc})}\,(\text{FULL–T–COMPLETE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash P_{ctx}[\sigma]\ \mathsf{t}^{a}$ By lemma truth sound

$\mathsf{True} \preccurlyeq \mathsf{t}^{a}$ By lemma truth sound

Let $\rho_c^{\Delta} = \mathtt{lattice}(\bar{Q}[\sigma], \mathcal{A}^{conc}, \mathcal{B}^{conc})$

By case analysis on t^{a}

Case: $\mathsf{t}^{a} = \mathsf{True}$

$(\Sigma_a^t, \Sigma_a^u) = \mathsf{allValidSubs}(\mathcal{A}^{abs}; \sigma; \mathsf{FV}(\mathtt{cons}))$ By lemma valid subs Sound and Complete

$\Sigma_c^t \subseteq \Sigma_a^t \cup \Sigma_a^u$ By lemma valid subs Sound and Complete

$\Sigma_c^u \subseteq \Sigma_a^u$ By lemma valid subs Sound and Complete

$\Sigma_c^t \supseteq \Sigma_a^t$ By lemma valid subs Sound and Complete

$\Sigma_c^t \cup \Sigma_c^u \subseteq \Sigma_a^t \cup \Sigma_a^u$ By subsets above

Let σ' where $\sigma' \in \Sigma_c^t \cup \Sigma_c^u$ and $\mathcal{B}^{conc}; \rho^{conc} \vdash P_{req}[\sigma']\ \mathsf{True} \;\vee\; \rho^{conc} \vdash P_{req}[\sigma']\ \mathsf{Unknown}$

$\sigma' \in \Sigma_a^t \cup \Sigma_a^u$ By $\Sigma_c^t \cup \Sigma_c^u \subseteq \Sigma_a^t \cup \Sigma_a^u$

$\mathcal{B}^{abs}; \rho^{abs} \vdash P_{req}[\sigma']\ \mathsf{True} \;\vee\; \mathcal{B}^{abs}; \rho^{abs} \vdash P_{req}[\sigma']\ \mathsf{Unknown}$ By lemma truth complete

Let $\rho_a^{\Delta} = \mathtt{lattice}(\bar{Q}[\sigma]; \mathcal{A}^{abs}; \mathcal{B}^{abs})$

$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} \mathtt{cons} \rightarrow \rho_a^{\Delta}$ By rule $\mathtt{full} - \mathsf{T} - \mathtt{sound}$

$\rho_c^{\Delta} \sqsubseteq \rho_a^{\Delta}$ By Lemma lattice complete

$\rho_c^{\Delta} \trianglelefteq \rho_a^{\Delta}$ By Lemma lattice complete

Case: $\mathsf{t}^{a} = \mathsf{False}$

Invalid case by $\mathsf{True} \preccurlyeq \mathsf{t}^{a}$

Case: $\mathsf{t}^{a} = \mathsf{Unknown}$

Let $\rho_{a}^{\Delta'} = \mathit{lattice}(\bar{Q}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})$
Let $\rho_{a}^{\Delta} = \updownarrow \rho^{\Delta'}$
$\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho_{a}^{\Delta}$ By rule $\mathit{full} - \mathit{complete} - \mathsf{Unknown}$
$\rho_{c}^{\Delta} \sqsubseteq \rho_{a}^{\Delta'}$ By Lemma lattice complete
$\rho_{c}^{\Delta} \trianglelefteq \rho_{a}^{\Delta'}$ By Lemma lattice complete
$\rho_{c}^{\Delta} \sqsubseteq \rho_{a}^{\Delta}$ By Lemma $\updownarrow$ on abs preserves $\sqsubseteq$
$\rho_{c}^{\Delta} \trianglelefteq \rho_{a}^{\Delta}$ By Lemma $\updownarrow$ on abs preserves $\trianglelefteq$

Case: $$\frac{\begin{array}{c}\mathit{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \\ \mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{Unknown} \qquad \rho_{c}^{\Delta} = \mathit{lattice}(\bar{Q}[\sigma])\end{array}}{\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \updownarrow \rho_{c}^{\Delta}}\ (\text{FULL–U–COMPLETE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{t}^{a}$ By lemma truth sound
$\mathsf{Unknown} \preccurlyeq \mathsf{t}^{a}$ By lemma truth sound
$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{Unknown}$ By inversion on $\mathsf{Unknown} \preccurlyeq \mathsf{t}^{a}$
Let $\rho_{a}^{\Delta} = \mathit{lattice}(\bar{Q}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})$
$\rho_{c}^{\Delta} \sqsubseteq \rho_{a}^{\Delta}$ By Lemma lattice complete
$\rho_{c}^{\Delta} \trianglelefteq \rho_{a}^{\Delta}$ By Lemma lattice complete
$\updownarrow \rho_{c}^{\Delta} \sqsubseteq \updownarrow \rho_{a}^{\Delta}$ By Lemma $\updownarrow$ preserves $\sqsubseteq$
$\updownarrow \rho_{c}^{\Delta} \trianglelefteq \updownarrow \rho_{a}^{\Delta}$ By Lemma $\updownarrow$ preserves $\trianglelefteq$

□

Theorem E.5. Truth Checking Complete

$$\begin{array}{l} \mathit{forall\ deriv.} \\ \qquad \rho^{conc} \sqsubseteq \rho^{abs} \\ \qquad \mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs} \\ \qquad \rho^{abs}\ \mathsf{final} \\ \qquad \rho^{conc}\ \mathsf{final} \\ \qquad \mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{P}[\sigma]\mathsf{t}^c \\ \mathit{exists\ deriv.} \\ \qquad \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}[\sigma]\mathsf{t}_a \\ \qquad \mathsf{t}^c \preccurlyeq \mathsf{t}_a \end{array}$$

Proof:
By induction on $\rho^{conc} \vdash \mathsf{P}[\sigma]\ \mathsf{t}_a$

Case: $\dfrac{\rho^{conc}(\mathsf{rel}(\bar{\ell})[\sigma]) = \mathtt{true}}{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{rel}(\bar{y})[\sigma]\ \mathsf{True}}(\text{REL}-\text{TRUE})$

Let $\mathsf{R} = \mathsf{rel}(\bar{\ell})[\sigma]$
$\mathsf{R} \in \mathit{dom}(\rho^{abs})$ By inversion on $\rho^{conc} \sqsubseteq \rho^{abs}$
Let $\mathsf{E}^a = \rho^{abs}(\mathsf{R})$
By case analysis on the value of E^a

Case: $\mathsf{E}^a = \mathtt{true}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{R}\ \mathsf{True}$ By rule rel $-$ True
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $\mathsf{E}^a = \mathtt{false}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $\mathsf{E}^a = \mathbf{unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{R}\ \mathsf{Unknown}$ By rule rel $-$ Unknown
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq$ $-$Unknown

Case: $\mathsf{E}^a = \mathtt{bot}$

Contradiction with ρ^{abs} final

Case: $\dfrac{\rho^{conc}(\mathsf{rel}(\bar{\ell})[\sigma]) = \mathtt{false}}{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{rel}(\bar{y})[\sigma]\ \mathsf{False}}(\text{REL}-\text{FALSE})$

Let $R = \mathsf{rel}(\bar{\ell})[\sigma]$
$R \in \mathrm{dom}(\rho^{abs})$ By inversion on $\rho^{conc} \sqsubseteq \rho^{abs}$
Let $E^{a} = \rho^{abs}(R)$
By case analysis on E^{a}

Case: $E^{a} = \mathtt{false}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash R\ \mathsf{True}$ By rule $\mathsf{rel} - \mathsf{False}$
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $E^{a} = \mathtt{true}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $E^{a} = \mathtt{unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash R\ \mathsf{Unknown}$ By rule $\mathsf{rel} - \mathsf{Unknown}$
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq - \mathsf{Unknown}$

Case: $E^{a} = \mathtt{bot}$

Contradiction with ρ^{abs} final

Case: $$\frac{\rho^{conc}(\mathsf{rel}(\bar{\ell})) = E^{c} \qquad E^{c} \neq \mathtt{true} \qquad E^{c} \neq \mathtt{false}}{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{rel}(\bar{\ell})\ \mathsf{Unknown}}\ (\mathsf{REL{-}UNKNOWN{-}SOUND{-}COMPLETE})$$

Let $R = \mathsf{rel}(\bar{\ell})[\sigma]$
$R \in \mathrm{dom}(\rho^{abs})$ By inversion on $\rho^{conc} \sqsubseteq \rho^{abs}$
Let $E^{a} = \rho^{abs}(R)$
By case analysis on E^{a}

Case: $E^{a} = \mathtt{false}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $E^{a} = \mathtt{true}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $E^{a} = \mathtt{unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash R\ \mathsf{Unknown}$ By rule $\mathsf{rel} - \mathsf{Unknown}$
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq - \mathsf{Unknown}$

Case: $E^{a} = \mathtt{bot}$

Contradiction with ρ^{abs} final

Case: $$\frac{\mathcal{B}^{conc}; \rho \vdash A\ t_c \qquad \mathcal{B}^{conc}(\ell_{test}) = t_c \qquad t^c \neq \mathsf{Unknown}}{\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{True}}\,(\mathsf{REL{-}TEST{-}TRUE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A\ t_a$ By induction hypothesis
$t^c \preccurlyeq t_a$ By induction hypothesis
By case analysis on t_c

Case: $t_c = \mathsf{True}$

By case analysis on $\mathcal{B}^{abs}(\ell_{test})$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{True}$

By case analysis on t_a

Case: $t_a = \mathsf{True}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{True}$ By rule $rel - test - \mathsf{True}$
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $t_a = \mathsf{False}$

Invalid case by $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $t_a = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $rel - test - \mathsf{Unknown1}$
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq$ $-\mathsf{Unknown}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{False}$

Invalid case by $\mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $rel - test - \mathsf{Unknown2}$

Case: $t_c = \mathsf{False}$

By case analysis on $\mathcal{B}^{abs}(\ell_{test})$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{False}$

By case analysis on t_a

Case: $t_a = \mathsf{False}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $rel - test - \mathsf{False}$
$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Case: $t_a = \mathsf{True}$

Invalid case by $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $t_a = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $\mathit{rel} - \mathit{test} - \mathsf{Unknown1}$

$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{Unknown}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{True}$

Invalid case by $\mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $\mathit{rel} - \mathit{test} - \mathsf{Unknown2}$

Case: $t_c = \mathsf{Unknown}$

Invalid case by $t_c \neq \mathsf{Unknown}$

Case: $$\frac{\mathcal{B}^{conc}; \rho \vdash A\ t_c^1 \qquad \mathcal{B}^{conc}(\ell_{test}) = t_c^2 \qquad t_1^c \neq \mathsf{Unknown}t_2^c \neq \mathsf{Unknown}t_1^c \neq t_2^c}{\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}}\ (\mathsf{REL{-}TEST{-}FALSE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A\ t_a^1$ By induction hypothesis

$t_c^1 \preccurlyeq t_a^1$ By induction hypothesis

By case analysis on t_c^1

Case: $t_c^1 = \mathsf{True}$

$t_c^2 = \mathsf{False}$ By $t_1^c \neq t_2^c$ and $t_1^c \neq \mathsf{Unknown}$

By case analysis on $\mathcal{B}^{abs}(\ell_{test})$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{False}$

By case analysis on t_a^1

Case: $t_a^1 = \mathsf{True}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $\mathit{rel} - \mathit{test} - \mathsf{False}$

$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Case: $t_a^1 = \mathsf{False}$

Invalid case by $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $t_a^1 = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $\mathit{rel} - \mathit{test} - \mathsf{Unknown1}$

$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{Unknown}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{True}$

Invalid case by $\mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $rel - test - \mathsf{Unknown2}$

Case: $t_c^1 = \mathsf{False}$

$t_c^2 = \mathsf{True}$ By $t_1^c \neq t_2^c$ and $t_1^c \neq \mathsf{Unknown}$

By case analysis on $\mathcal{B}^{abs}(\ell_{test})$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{True}$

By case analysis on t_a^1

Case: $t_a^1 = \mathsf{False}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $rel - test - \mathsf{False}$

$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Case: $t_a^1 = \mathsf{True}$

Invalid case by $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $t_a^1 = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $rel - test - \mathsf{Unknown1}$

$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{Unknown}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{False}$

Invalid case by $\mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}$

Case: $\mathcal{B}^{abs}(\ell_{test}) = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $rel - test - \mathsf{Unknown2}$

Case: $t_c^1 = \mathsf{Unknown}$

Invalid case by $t_c \neq \mathsf{Unknown}$

Case: $$\frac{\mathcal{B}^{conc}; \rho^{conc} \vdash A\ \mathsf{Unknown}}{\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{Unknown}}\ (\mathsf{REL{-}TEST{-}U1})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A\ t_a$ By induction hypothesis

$\mathsf{Unknown} \preccurlyeq t_a$ By induction hypothesis

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $rel - test - u1$

$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{Unknown}$

Case: $$\frac{\mathcal{B}^{conc}(\ell_{test}) = \mathsf{Unknown} \qquad \mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{A}\ t_c}{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{A}/\ell_{test}\ \mathsf{Unknown}}\,(\text{REL—TEST—U2})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{A}\ t_a$	By induction hypothesis
$t_c \preccurlyeq t_a$	By induction hypothesis
$\mathcal{B}^{abs}(\ell_{test}) = \mathsf{Unknown}$	By $\mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}$
$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{A}/\ell_{test}\ \mathsf{Unknown}$	By rule $rel - test - u2$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$	By rule $\preccurlyeq -\mathsf{Unknown}$

Case: $$\frac{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{S}\ \mathsf{Unknown}}{\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{S}\ \mathsf{Unknown}}\,(\neg\text{S—UNKNOWN})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{S}\ t_a$	By induction hypothesis
$\mathsf{Unknown} \preccurlyeq t_a$	By induction hypothesis
$\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\ \mathsf{Unknown}$	By rule $\neg\mathsf{S} - \mathsf{Unknown}$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$	By rule $\preccurlyeq -\mathsf{Unknown}$

Case: $$\frac{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{SFalse}}{\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{STrue}}\,(\neg\text{S—TRUE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{S}\ t_a$	By induction hypothesis
$\mathsf{False} \preccurlyeq t_a$	By induction hypothesis

By case analysis on the value of t_a

Case: $t_a = \mathsf{False}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\ \mathsf{True}$	By rule $\neg\mathsf{S} - \mathsf{True}$
$\mathsf{True} \preccurlyeq \mathsf{True}$	By rule $\preccurlyeq - =$

Case: $t_a = \mathsf{True}$

Contradiction with $\mathsf{False} \preccurlyeq t_a$

Case: $t_a = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\ \mathsf{Unknown}$	By rule $\neg\mathsf{S} - \mathsf{Unknown}$
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$	By rule $\preccurlyeq - =$

Case: $$\frac{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{STrue}}{\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{SFalse}}\,(\neg\text{R—FALSE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{S}\ t_a$	By induction hypothesis
$\mathsf{True} \preccurlyeq t_a$	By induction hypothesis

By case analysis on the value of t_a

Case: $t_a = \mathsf{True}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\ \mathsf{False}$ By rule $\neg\mathsf{S} - \mathsf{False}$
$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Case: $t_a = \mathsf{False}$

Contradiction with $\mathsf{True} \preccurlyeq t_a$

Case: $t_a = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\ \mathsf{Unknown}$ By rule $\neg\mathsf{S} - \mathsf{Unknown}$
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq - =$

Case: $$\frac{}{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathit{true}\mathsf{True}}(\text{TRUE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathit{true}\mathsf{True}$ By rule true
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $$\frac{}{\mathcal{B}^{conc}; \rho^{conc} \vdash \mathit{false}\mathsf{False}}(\text{FALSE})$$

$\mathcal{B}^{abs}; \rho^{abs} \vdash \mathit{false}\mathsf{False}$ By rule false
$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Remaining cases work as expected for a three value logic.

□

Theorem E.6. Instruction Binding Complete

$$\begin{array}{l} \mathit{forall\ deriv.} \\ \qquad \mathcal{A}^{\mathrm{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathrm{abs}} \\ \qquad \mathcal{A}^{\mathrm{conc}} \vdash \mathit{instr} : \mathsf{op} \hookrightarrow (\Sigma^t_c, \Sigma^u_c) \\ \mathit{exists\ deriv.} \\ \qquad \mathcal{A}^{\mathrm{abs}} \vdash \mathit{instr} : \mathsf{op} \hookrightarrow (\Sigma^t_a, \Sigma^u_a) \\ \qquad \Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a \\ \qquad \Sigma^u_c \subseteq \Sigma^u_a \\ \qquad \Sigma^t_c \supseteq \Sigma^t_a \end{array}$$

Proof:
By case analysis on the structure of the derivation of $\mathcal{A}^{\mathrm{conc}} \vdash \mathit{instr} : \mathsf{op} \hookrightarrow (\Sigma^t_c, \Sigma^u_c)$

Case: $$\frac{\begin{array}{c} \mathsf{FV}(\tau_{\mathsf{this}}.\mathsf{m}(\overline{\mathsf{y}} : \overline{\tau}) : \tau_{\mathsf{ret}}) \subseteq \Gamma_{\mathsf{y}} \\ (\Sigma^t_c, \Sigma^u_c) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{abs}}, \Gamma_{\mathsf{y}}, \{\mathtt{x_{ret}}, \mathtt{x_{this}}\} \cup \overline{\mathtt{x}}, \{\mathsf{ret}, \mathsf{this}\} \cup \overline{\mathsf{y}}) \end{array}}{\mathcal{A}^{\mathrm{conc}}; \Gamma_{\mathsf{y}} \vdash \mathtt{x_{ret}} = \mathtt{x_{this}}.\mathsf{m}(\overline{\mathtt{x}}) : \tau_{\mathsf{this}}.\mathsf{m}(\overline{\mathsf{y}} : \overline{\tau}) : \tau_{\mathsf{ret}} \mapsto (\Sigma^t_c, \Sigma^u_c)}\,(\text{INVOKE})$$

$(\Sigma^t_a, \Sigma^u_a) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{abs}}, \Gamma_{\mathsf{y}}, \{\mathtt{x_{ret}}, \mathtt{x_{this}}\} \cup \overline{\mathtt{x}}, \{\mathsf{ret}, \mathsf{this}\} \cup \overline{\mathsf{y}})$By lemma FindLabels sound and complete
$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By lemma FindLabels sound and complete
$\Sigma^u_c \subseteq \Sigma^u_a$ By lemma FindLabels sound and complete
$\Sigma^t_c \supseteq \Sigma^t_a$ By lemma FindLabels sound and complete
$\mathcal{A}^{\mathrm{abs}} \vdash \mathtt{x_{ret}} = \mathtt{x_{this}}.\mathsf{m}(\overline{\mathtt{x}}) : \tau_{\mathsf{this}}.\mathsf{m}(\overline{\mathsf{y}} : \overline{\tau}) : \tau_{\mathsf{ret}} \hookrightarrow (\Sigma^t_a, \Sigma^u_a)$ By rule invoke

Case: $$\frac{\begin{array}{c} \mathsf{FV}(\ \mathsf{new}\ \ \tau(\overline{\mathsf{y}} : \overline{\tau})) \subseteq \Gamma_{\mathsf{y}} \\ (\Sigma^t_c, \Sigma^u_c) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{conc}}, \Gamma_{\mathsf{y}}, \{\mathtt{x_{ret}}\} \cup \overline{\mathtt{x}}, \{\mathsf{this}\} \cup \overline{\mathsf{y}}) \end{array}}{\mathcal{A}^{\mathrm{conc}}; \Gamma_{\mathsf{y}} \vdash \mathtt{x_{ret}} = \ \mathtt{new}\ \ \mathrm{m}(\overline{\mathtt{x}}) :\ \mathsf{new}\ \ \tau(\overline{\mathsf{y}} : \overline{\tau}) \mapsto (\Sigma^t_c, \Sigma^u_c)}\,(\text{CONSTRUCTOR})$$

$(\Sigma^t_a, \Sigma^u_a) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{abs}}, \Gamma_{\mathsf{y}}, \{\mathtt{x_{ret}}, \mathtt{x_{this}}\} \cup \overline{\mathtt{x}}, \{\mathsf{ret}, \mathsf{this}\} \cup \overline{\mathsf{y}})$By lemma FindLabels sound and complete
$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By lemma FindLabels sound and complete
$\Sigma^u_c \subseteq \Sigma^u_a$ By lemma FindLabels sound and complete
$\Sigma^t_c \supseteq \Sigma^t_a$ By lemma FindLabels sound and complete
$\mathcal{A}^{\mathrm{abs}} \vdash \mathtt{x_{ret}} = \ \mathtt{new}\ \ \mathrm{m}(\overline{\mathtt{x}}) :\ \mathsf{new}\ \ \tau(\overline{\mathsf{y}} : \overline{\tau}) \hookrightarrow (\Sigma^t_a, \Sigma^u_a)$ By rule $\mathit{constructor}$

Case: $$\frac{}{\mathcal{A}^{\mathrm{conc}}; \Gamma_{\mathsf{y}} \vdash \mathit{eom} : \mathsf{end\text{-}of\text{-}method} \mapsto (\{\varnothing\}, \varnothing)}\,(\text{EOM})$$

$\mathcal{A}^{\mathrm{abs}} \vdash \mathit{eom} : \mathsf{end\text{-}of\text{-}method} \mapsto (\{\varnothing\}, \varnothing)$ By rule eom
$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By $\{\varnothing\} \subseteq \{\varnothing\} \cup \varnothing$
$\Sigma^u_c \subseteq \Sigma^u_a$ By $\varnothing \subseteq \varnothing$
$\Sigma^t_c \supseteq \Sigma^t_a$ By $\{\varnothing\} \supseteq \{\varnothing\}$

□

F Soundness

Theorem F.1. Soundness of Relations Analysis

$$
\begin{array}{l}
\mathsf{forall\ der.} \\
\qquad f_{\mathsf{alias}}(\mathcal{A}^{\mathsf{abs}}, \mathsf{instr}) = \mathcal{A}^{\mathsf{abs}'} \\
\qquad f_{\mathsf{alias}}(\mathcal{A}^{\mathsf{conc}}, \mathsf{instr}) = \mathcal{A}^{\mathsf{conc}'} \\
\qquad \rho^{\mathsf{abs}}\ \mathsf{final} \\
\qquad \rho^{\mathsf{conc}}\ \mathsf{final} \\
\qquad \mathcal{B}^{\mathsf{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathsf{abs}} \\
\qquad \mathcal{A}^{\mathsf{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathsf{abs}} \\
\qquad \mathcal{A}^{\mathsf{abs}} \vdash \rho^{\mathsf{abs}}\ \mathsf{consistent} \\
\qquad \mathcal{A}^{\mathsf{conc}} \vdash \rho^{\mathsf{conc}}\ \mathsf{consistent} \\
\qquad \rho^{\mathsf{conc}} \sqsubseteq \rho^{\mathsf{abs}} \\
\qquad f_{\mathcal{C};\mathcal{A}^{\mathsf{abs}};\mathcal{B}^{\mathsf{abs}}}(\rho^{\mathsf{abs}}, \mathsf{instr}) = \rho^{\mathsf{abs}'} \\
\mathsf{exists\ der.} \\
\qquad f_{\mathcal{C};\mathcal{A}^{\mathsf{conc}};\mathcal{B}^{\mathsf{conc}}}(\rho^{\mathsf{conc}}, \mathsf{instr}) = \rho^{\mathsf{conc}'} \\
\qquad \rho^{\mathsf{conc}'} \sqsubseteq \rho^{\mathsf{abs}'}
\end{array}
$$

Proof: [Soundness of Relation Analysis]

$\rho^{\mathsf{abs}'} = \mathsf{transfer}(\rho^{\mathsf{abs}}, \mathcal{A}^{\mathsf{abs}'}) \boxed{\leftarrow} \rho^{\mathsf{abs}\Delta}$	By inversion on $f_{\mathcal{C};\mathcal{A}^{\mathsf{abs}};\mathcal{B}^{\mathsf{abs}}}(\rho^{\mathsf{abs}}, \mathsf{instr}) = \rho^{\mathsf{abs}'}$
$\forall\, \mathsf{cons}_i \in \mathcal{C}\ .\ \mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}\rho^{\mathsf{abs}}; \mathsf{cons}_i \vdash \mathsf{instr} \hookrightarrow \rho_i^{\mathsf{abs}\Delta}$	By inversion on $f_{\mathcal{C};\mathcal{A}^{\mathsf{abs}};\mathcal{B}^{\mathsf{abs}}}(\rho^{\mathsf{abs}}, \mathsf{instr}) = \rho^{\mathsf{abs}'}$
$\rho^{\mathsf{abs}\Delta} = \sqcup \{\rho_i^{\mathsf{abs}\Delta}\}$	By inversion on $f_{\mathcal{C};\mathcal{A}^{\mathsf{abs}};\mathcal{B}^{\mathsf{abs}}}(\rho^{\mathsf{abs}}, \mathsf{instr}) = \rho^{\mathsf{abs}'}$
$\forall\, \mathsf{cons}_i \in \mathcal{C}\ .$	
$\quad \rho_i^{\mathsf{conc}\Delta} \sqsubseteq \rho_i^{\mathsf{abs}\Delta}$	By Lemma Soundness of Single Constraint
$\quad \mathcal{A}^{\mathsf{conc}'}; \rho^{\mathsf{conc}}; \mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \rho_i^{\mathsf{conc}\Delta}$	By Lemma Soundness of Single Constraint
$\quad \rho_i^{\mathsf{conc}\Delta} \trianglelefteq \rho_i^{\mathsf{abs}\Delta}$	By Lemma Soundness of Single Constraint
$\quad \mathcal{A}^{\mathsf{conc}'} \vdash \rho_i^{\mathsf{conc}\Delta}\ \mathsf{consistent}$	By Lemma Consistency of Single Constraint
$\exists\, \bar{R}\ .\ \forall\, i\ .\ \mathsf{dom}(\rho_i^{\mathsf{conc}\Delta}) = \bar{R}$	By Lemma consistency means same domain
Let $\rho^{\mathsf{conc}\Delta} = \sqcup \{\rho_i^{\mathsf{conc}\Delta}\}$	By join rule applied many times
$\rho^{\mathsf{conc}\Delta} \trianglelefteq \rho^{\mathsf{abs}\Delta}$	By Lemma $\sqcup$ preserves $\trianglelefteq$
$\rho^{\mathsf{conc}\Delta} \sqsubseteq \rho^{\mathsf{abs}\Delta}$	By Lemma $\sqcup$ preserves $\sqsubseteq$
$\mathcal{A}^{\mathsf{conc}'} \vdash \rho^{\mathsf{conc}\Delta}\ \mathsf{consistent}$	By Lemma same domains mean consistency
Let $\rho^{\mathsf{conc}''} = \mathsf{transfer}(\rho^{\mathsf{conc}}, \mathcal{A}^{\mathsf{conc}'})$	
$\mathcal{A}^{\mathsf{conc}'} \vdash \rho^{\mathsf{conc}''}\ \mathsf{consistent}$	By Lemma transfer implies consistency
$\mathsf{dom}(\rho^{\mathsf{conc}''}) = \mathsf{dom}(\rho^{\mathsf{conc}\Delta})$	By Lemma consistency means same domain
Let $\rho^{\mathsf{conc}'} = \rho^{\mathsf{conc}''} \boxed{\leftarrow} \rho^{\mathsf{conc}\Delta}$	By rule overmeets
$\rho^{\mathsf{conc}'} \sqsubseteq \rho^{\mathsf{abs}'}$	By Lemma $\boxed{\leftarrow}$ preserves $\sqsubseteq$

$f_{\mathcal{C};\mathcal{A}^{\mathrm{conc}};\mathcal{B}^{\mathrm{conc}}}(\rho^{\mathrm{conc}}, \mathit{instr}) = \rho^{\mathrm{conc}'}$ By rule $flow - cons$

□

Theorem F.2. Soundness of Single Constraint

$$
\begin{array}{l}
\mathit{forall\ deriv.} \\
\qquad \mathcal{A}^{\mathsf{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathsf{abs}} \\
\qquad \mathcal{B}^{\mathsf{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathsf{abs}} \\
\qquad \rho^{\mathsf{conc}} \sqsubseteq \rho^{\mathsf{abs}} \\
\qquad \mathcal{A}^{\mathsf{abs}} \vdash \rho^{\mathsf{abs}} \ \textsf{consistent} \\
\qquad \mathcal{A}^{\mathsf{conc}} \vdash \rho^{\mathsf{conc}} \ \textsf{consistent} \\
\qquad \rho^{\mathsf{conc}} \ \textsf{final} \\
\qquad \mathcal{A}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \rho^{\mathsf{abs}\Delta} \\
\mathit{exists\ deriv.} \\
\qquad \mathcal{A}^{\mathsf{conc}}; \rho^{\mathsf{conc}}; \mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \rho^{\mathsf{conc}\Delta} \\
\qquad \rho^{\mathsf{conc}\Delta} \sqsubseteq \rho^{\mathsf{abs}\Delta} \\
\qquad \rho^{\mathsf{conc}\Delta} \trianglelefteq \rho^{\mathsf{abs}\Delta}
\end{array}
$$

Proof:
By case analysis on $\mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \rho^{\mathsf{abs}\Delta}$

Case:

$$
\frac{\begin{array}{c}
\mathsf{cons} = \mathsf{op} : \mathsf{P}_{\mathsf{ctx}} \Rightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{A}^{\mathsf{abs}}; \mathsf{FV}(\mathsf{cons}) \vdash \mathsf{instr} : \mathsf{op} \Mapsto (\Sigma^{\mathsf{t}}_{\mathsf{a}}, \Sigma^{\mathsf{u}}_{\mathsf{a}}) \\
\Sigma^{\mathsf{t}}_{\mathsf{a}} \neq \varnothing \vee \Sigma^{\mathsf{u}}_{\mathsf{a}} \neq \varnothing \qquad \mathcal{P}^{\mathsf{t}}_{\mathsf{a}} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{\mathsf{t}}_{\mathsf{a}} \wedge \mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\
\mathcal{P}^{\mathsf{u}}_{\mathsf{a}} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{\mathsf{u}}_{\mathsf{a}} \wedge \mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\} \\
|\mathcal{P}^{\mathsf{t}}_{\mathsf{a}}| = |\Sigma^{\mathsf{t}}_{\mathsf{a}}| \qquad |\mathcal{P}^{\mathsf{u}}_{\mathsf{a}}| = |\Sigma^{\mathsf{u}}_{\mathsf{a}}| \qquad \mathcal{P}^{\Delta}_{\mathsf{a}} = \mathcal{P}^{\mathsf{t}}_{\mathsf{a}} \cup \mathcal{P}^{\mathsf{u}}_{\mathsf{a}}
\end{array}}{\mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho\mathcal{B}^{\mathsf{abs}}; \mathsf{cons} \vdash \mathsf{instr} \hookrightarrow (\bigsqcup \mathcal{P}^{\Delta}_{\mathsf{a}})} \ (\mathsf{MATCH})
$$

Let $\rho^{\Delta}_{\mathsf{a}} = (\bigsqcup \mathcal{P}^{\Delta}_{\mathsf{a}})$

$\mathcal{A}^{\mathsf{conc}} \vdash \mathsf{instr} : \mathsf{op} \Mapsto (\Sigma^{\mathsf{t}}_{\mathsf{c}}, \Sigma^{\mathsf{t}}_{\mathsf{a}})$	By Lemma Instruction Binding Sound
$\Sigma^{\mathsf{t}}_{\mathsf{c}} \subseteq \Sigma^{\mathsf{t}}_{\mathsf{a}} \cup \Sigma^{\mathsf{u}}_{\mathsf{a}}$	By lemma Instruction Binding Sound
$\Sigma^{\mathsf{u}}_{\mathsf{c}} \subseteq \Sigma^{\mathsf{u}}_{\mathsf{a}}$	By lemma Instruction Binding Sound
$\Sigma^{\mathsf{t}}_{\mathsf{c}} \supseteq \Sigma^{\mathsf{t}}_{\mathsf{a}}$	By lemma Instruction Binding Sound

By case analysis on the property $\Sigma^{\mathsf{t}}_{\mathsf{c}} \cup \Sigma^{\mathsf{u}}_{\mathsf{c}} = \varnothing$

Case: $\Sigma^{\mathsf{t}}_{\mathsf{c}} \cup \Sigma^{\mathsf{u}}_{\mathsf{c}} = \varnothing$

$\Sigma^{\mathsf{t}}_{\mathsf{c}} = \varnothing$	By inversion of $\Sigma^{\mathsf{t}}_{\mathsf{c}} \cup \Sigma^{\mathsf{u}}_{\mathsf{c}} = \varnothing$
$\Sigma^{\mathsf{u}}_{\mathsf{c}} = \varnothing$	By inversion of $\Sigma^{\mathsf{t}}_{\mathsf{c}} \cup \Sigma^{\mathsf{u}}_{\mathsf{c}} = \varnothing$
$\mathcal{A}^{\mathsf{conc}}; \mathcal{B}^{\mathsf{conc}}; \rho^{\mathsf{conc}}; \mathsf{cons} \vdash \mathsf{instr} \hookrightarrow \bot^{\mathsf{conc}}_{\mathcal{A}}$	By rule $\mathsf{not} - \mathsf{match}$
$\mathcal{A}^{\mathsf{conc}} \vdash \bot^{\mathsf{conc}}_{\mathcal{A}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$\mathcal{A}^{\mathsf{abs}} \vdash \rho^{\Delta}_{\mathsf{a}}$ consistent	By Lemma partial binding consistent
$\mathsf{dom}(\bot^{\mathsf{conc}}_{\mathcal{A}}) \subseteq \mathsf{dom}(\rho^{\Delta}_{\mathsf{a}})$	By lemma consistency and $\sqsubseteq_{\mathcal{A}}$ implies ρ domains subset

$\forall\, \mathsf{R} \mapsto \mathsf{E}_{\mathsf{c}} \in \bot^{\mathsf{conc}}_{\mathcal{A}}$.

$E_c = \mathtt{bot}$ By definition of $\bot_{\mathcal{A}}$

$E_c \sqsubseteq \rho^{\Delta}_{a}(R)$ By rule $\sqsubseteq -\mathtt{bot}$

$\forall\, R \mapsto E_c \in \bot^{conc}_{\mathcal{A}} \,.\, E_c \sqsubseteq \rho^{\Delta}_{a}(R)$ By quantification above

$\bot^{conc}_{\mathcal{A}} \sqsubseteq \rho^{\Delta}_{a}(R)$ By rule $\sqsubseteq -\rho$

$\Sigma^{t}_{a} = \varnothing$ By $\Sigma^{t}_{c} \supseteq \Sigma^{t}_{a}$ and $\Sigma^{t}_{c} = \varnothing$

$\mathcal{P}^{t}_{a} = \varnothing$ By $|\mathcal{P}^{t}_{a}| = |\Sigma^{t}_{a}|$

$\forall\, \rho^{u\Delta}_{a} \in \mathcal{P}^{u}_{a} \,.$

Let $\rho^{u\Delta}_{a} =\!\updownarrow \rho^{u\Delta'}_{a}$

Where $\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^{u} \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{u\Delta'}_{a}$ and $\sigma^{u} \in \Sigma^{u}_{a}$ By construction of $\mathcal{P}^{u}_{a}$

$\forall\, R \mapsto E \in \rho^{u\Delta}_{a} \,.\, E = \mathtt{bot} \vee E = \mathsf{unknown}$ By $\updownarrow$ makes everything bottom or top.

$\forall\, \rho^{u\Delta}_{a} \in \mathcal{P}^{u}_{a} \,.\, \forall\, R \mapsto E \in \rho^{u\Delta}_{a} \,.\, E = \mathtt{bot} \vee E = \mathsf{unknown}$ By quantification

$\forall\, R \mapsto E \in \rho^{\Delta}_{a} \,.\, E = \mathtt{bot} \vee E = \mathsf{unknown}$ By $\sqcup\!\!\!\!=$ preserves polarity

$\forall\, R \in \mathsf{dom}(\bot_{\mathcal{A}^{conc}}) \,.$

$\bot_{\mathcal{A}^{conc}}(R) = \mathtt{bot}$ By definition of $\bot$

Let $E_a = \rho^{\Delta}_{a}(R)$

Case analysis on the value of E_a

$E_a = \mathtt{bot}$

$\mathtt{bot} \trianglelefteq \mathtt{bot}$ By rule $\trianglelefteq - \mathtt{bot}$

$E_a = \mathsf{unknown}$

$\mathtt{bot} \trianglelefteq \mathsf{unknown}$ By rule $\trianglelefteq - \mathsf{unknown}$

$E_a = \mathtt{true}$

Contradiction with $\forall\, R \mapsto E \in \rho^{\Delta}_{a} \,.\, E = \mathtt{bot} \vee E = \mathsf{unknown}$

$E_a = \mathtt{false}$

Contradiction with $\forall\, R \mapsto E \in \rho^{\Delta}_{a} \,.\, E = \mathtt{bot} \vee E = \mathsf{unknown}$

$\forall\, R \in \mathsf{dom}(\bot_{\mathcal{A}^{conc}}) \,.\, \bot_{\mathcal{A}^{conc}}(R) \trianglelefteq \rho^{\Delta}_{a}(R)$ By quantification

$\bot_{\mathcal{A}^{conc}} \trianglelefteq \rho^{\Delta}_{a}$ By rule $\trianglelefteq - \rho$

Case: $\Sigma^{t}_{c} \cup \Sigma^{u}_{c} \neq \varnothing$

$\Sigma^{t}_{c} \neq \varnothing \vee \Sigma^{u}_{c} \neq \varnothing$ By inversion on $\cup$

Let $\mathcal{P}^{t}_{c} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{t}_{c} \ \wedge\ \mathcal{A}^{conc}; \mathcal{B}; \rho^{conc}; \sigma \vdash_{\mathsf{part}} \mathsf{cons} \hookrightarrow \rho^{\Delta}\}\ \forall\, \sigma^{t} \in \Sigma^{t}_{c} \,.$

$\sigma^t \in \Sigma_a^t \cup \Sigma_a^u$ By inversion on $\Sigma_c^t \subseteq \Sigma_a^t \cup \Sigma_a^u$

Case analysis on the location of σ^t

$\sigma^t \in \Sigma_a^t$

$\exists$ distinct $\rho_a^{t\Delta} \in \mathcal{P}_a^t \,.\, \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^t \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_a^{t\Delta}$

By the construction of $\mathcal{P}_a^t$ and $|\mathcal{P}_a^t| = |\Sigma_a^t|$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^t \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_c^{t\Delta}$By lemma partial constraint binding sound

$\mathcal{A}_{conc} \vdash \rho_c^{t\Delta}$ consistent By lemma partial constraint consistent

$\rho_c^{t\Delta} \sqsubseteq \rho_a^{t\Delta}$ By lemma partial constraint binding sound

$\rho_c^{t\Delta} \trianglelefteq \rho_a^{t\Delta}$ By lemma partial constraint binding sound

$\rho_c^{t\Delta} \in \mathcal{P}_c^t$ By construction of $\mathcal{P}_c^t$

$\sigma^t \in \Sigma_a^u$

$\exists$ distinct $\rho_a^{u\Delta} \in \mathcal{P}_a^u$.

$\rho_a^{u\Delta} =\updownarrow \rho_a^{u\Delta'} \ \wedge\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^t \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_a^{u\Delta'}$

By the construction of $\mathcal{P}_a^u$ and $|\mathcal{P}_a^u| = |\Sigma_a^u|$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^t \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_c^{t\Delta}$By lemma partial constraint binding sound

$\mathcal{A}_{conc} \vdash \rho_c^{t\Delta}$ consistent By lemma partial constraint consistent

$\rho_c^{t\Delta} \sqsubseteq \rho_a^{u\Delta'}$ By lemma partial constraint binding sound

$\rho_c^{t\Delta} \trianglelefteq \rho_a^{u\Delta'}$ By lemma partial constraint binding sound

$\rho_c^{t\Delta} \sqsubseteq \rho_a^{u\Delta}$ By lemma $\updownarrow$ on right preserves $\sqsubseteq$

$\rho_c^{t\Delta} \trianglelefteq \rho_a^{u\Delta}$ By lemma $\updownarrow$ on right preserves $\trianglelefteq$

$\rho_c^{t\Delta} \in \mathcal{P}_c^t$ By construction of $\mathcal{P}_c^u$

$\forall\, \sigma^t \in \Sigma_c^t \,.\, \exists$ distinct $\rho_c^t \in \mathcal{P}_c^t \,.\, \mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^t \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_c^t$ By quantification above

$|\mathcal{P}_c^t| = |\Sigma_c^t|$ By quantification above and construction of $\mathcal{P}_c^t$

$\forall\, \rho_c^{t\Delta} \in \mathcal{P}_c^t \,.\, \exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a \,.\, \rho_c^{t\Delta} \sqsubseteq \rho_a^{\Delta}$ By quantification above

$\forall\, \rho_c^{t\Delta} \in \mathcal{P}_c^t \,.\, \exists$ distinct $\rho_a^{\Delta} \in \mathcal{P}_a \,.\, \rho_c^{t\Delta} \trianglelefteq \rho_a^{\Delta}$ By quantification above

$\forall\, \rho_c^{t\Delta} \in \mathcal{P}_c^t \,.\, \mathcal{A}_{conc} \vdash \rho_c^{t\Delta}$ consistent By quantification above

Let $\mathcal{P}_c^u = \{\rho^\Delta \mid \sigma \in \Sigma_c^u \ \wedge\ \mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho^\Delta\}$

$\forall\, \sigma^u \in \Sigma_c^u$.

$\sigma^u \in \Sigma_a^u$ By inversion on $\Sigma_c^u \subseteq \Sigma_a^u$

$\exists$ distinct $\rho_a^{u\Delta'} \in \mathcal{P}_a^u \,.\, \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^u \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_a^{u\Delta'}$

By the construction of $\mathcal{P}_a^u$ and $|\mathcal{P}_a^u| = |\Sigma_a^u|$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^u \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow \rho_c^{u\Delta'}$ By lemma partial constraint binding sound

$\mathcal{A}_{conc} \vdash \rho_c^{u\Delta'}$ consistent By lemma partial constraint consistent

Let $\rho_a^{u\Delta} =\updownarrow \rho_a^{u\Delta'}$

$\mathrm{dom}(\rho_c^{u\Delta'}) = \mathrm{dom}(\rho^{conc})$ By lemma consistency implies same domain

Let $\rho_c^{u\Delta} =\updownarrow \rho_c^{u\Delta'}$

$\rho_c^{u\Delta'} \sqsubseteq \rho_a^{u\Delta'}$ By lemma partial constraint binding sound

$\rho_c^{u\Delta'} \trianglelefteq \rho_a^{u\Delta'}$ By lemma partial constraint binding sound

$\rho_c^{u\Delta} \sqsubseteq \rho_a^{u\Delta}$ By lemma $\updownarrow$ preserves $\sqsubseteq$

$\rho_c^{u\Delta} \trianglelefteq \rho_a^{u\Delta}$ By lemma $\updownarrow$ preserves $\trianglelefteq$

$\rho_c^{u\Delta} \in \mathcal{P}_c^u$ By construction of $\mathcal{P}_c^u$

$\forall\, \sigma^u \in \Sigma_c^u \,.\, \exists\, \textsf{distinct}\ \rho_c^u \in \mathcal{P}_c^u \,.\, \mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^u \vdash_{\textsf{part}} cons \hookrightarrow \rho_c^u$ By quantification above

$|\mathcal{P}_c^u| = |\Sigma_c^u|$ By quantification above and construction of $\mathcal{P}_c^t$

$\forall\, \rho_c^{u\Delta} \in \mathcal{P}_c^u \,.\, \exists\, \textsf{distinct}\ \rho_a^\Delta \in \mathcal{P}_a \,.\, \rho_c^{u\Delta} \sqsubseteq \rho_a^\Delta$ By quantification above

$\forall\, \rho_c^{u\Delta} \in \mathcal{P}_c^u \,.\, \exists\, \textsf{distinct}\ \rho_a^\Delta \in \mathcal{P}_a \,.\, \rho_c^{u\Delta} \trianglelefteq \rho_a^\Delta$ By quantification above

$\forall\, \rho_c^{u\Delta} \in \mathcal{P}_c^u \,.\, \mathcal{A}_{conc} \vdash \rho_c^{u\Delta}\ \textsf{consistent}$ By quantification above

Let $\mathcal{P}_c = \mathcal{P}_c^t \cup \mathcal{P}_c^u$

$\exists \bar{R} \,.\, \forall\, \rho_c \in \mathcal{P}_c \,.\, \mathrm{dom}(\rho_c) = \bar{R}$ By inversion on consistency of each $\rho_c^{t\Delta}$

Let $\rho_c^\Delta = (\, \boxminus\, \mathcal{P}_c)$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; cons \vdash instr \hookrightarrow \rho^{conc\Delta}$ By rule $match$

$\forall\, \rho_c^\Delta \in \mathcal{P}_c \,.\, \exists\, \textsf{distinct}\ \rho_a^\Delta \in \mathcal{P}_a \,.\, \rho_c^\Delta \sqsubseteq \rho_a^\Delta$ By $\mathcal{P}_c = \textsf{Rho}_c^t \cup \textsf{Rho}_c^u$

$\forall\, \rho_c^\Delta \in \mathcal{P}_c \,.\, \exists\, \textsf{distinct}\ \rho_a^\Delta \in \mathcal{P}_a \,.\, \rho_c^\Delta \trianglelefteq \rho_a^\Delta$ By $\mathcal{P}_c = \textsf{Rho}_c^t \cup \textsf{Rho}_c^u$

$\rho_c^\Delta \sqsubseteq \rho_a^\Delta$ By $\boxminus$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

$\rho_c^\Delta \trianglelefteq \rho_a^\Delta$ By $\boxminus$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

Case: $$\frac{cons = \textsf{op} : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \qquad \mathcal{A}^{abs}; \textsf{FV}(cons) \vdash instr : \textsf{op} \mapsto (\varnothing, \varnothing)}{\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; cons \vdash instr \hookrightarrow \bot_{\mathcal{A}^{abs}}}\ (\textsf{NO–MATCH})$$

$\mathcal{A}^{conc} \vdash instr : \textsf{op} \mapsto (\Sigma_c^t, \Sigma_a^t)$ By Lemma Instruction Binding Sound

$\Sigma_c^t \subseteq \Sigma_a^t \cup \Sigma_a^u$ By lemma Instruction Binding Sound

$\Sigma_c^t = \varnothing$ By inversion on $\subseteq$

$\Sigma_c^u \subseteq \Sigma_a^u$ By lemma Instruction Binding Sound

$\Sigma_c^u = \varnothing$ By inversion on $\subseteq$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; cons \vdash instr \hookrightarrow \bot_{\mathcal{A}^{conc}}$ By rule $not - match$

$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}}^{conc}\ \textsf{consistent}$ By definition of $\bot_{\mathcal{A}}$

$\forall\, R \in \mathrm{dom}(\bot_{\mathcal{A}}^{conc}) \,.\, \bot_{\mathcal{A}}^{conc}(R) = \texttt{bot}$ By definition of $\bot_{\mathcal{A}}$

$\mathcal{A}^{abs} \vdash \bot_{\mathcal{A}}^{abs}\ \textsf{consistent}$ By definition of $\bot_{\mathcal{A}}$

$\forall\, R \in \mathrm{dom}(\bot_{\mathcal{A}}^{abs}) \,.\, \bot_{\mathcal{A}}^{abs}(R) = \texttt{bot}$ By definition of $\bot_{\mathcal{A}}$

$\mathrm{dom}(\bot_{\mathcal{A}}^{conc}) \subseteq \mathrm{dom}(\bot_{\mathcal{A}}^{abs}$ By lemma consistency and $\sqsubseteq_{\mathcal{A}}$ implies ρ domains subset

$\forall\, R \in \mathrm{dom}(\bot_{\mathcal{A}}^{conc}) \,.\, \bot_{\mathcal{A}}^{conc}(R) \sqsubseteq \bot_{\mathcal{A}}^{abs}(R)$ By rule $\sqsubseteq -\texttt{bot}$

$\bot_{\mathcal{A}}^{conc} \sqsubseteq \bot_{\mathcal{A}}^{abs}$ By rule $\sqsubseteq -\rho$

$\forall\, R \in \mathrm{dom}(\bot_{\mathcal{A}}^{conc}) \,.\, \bot_{\mathcal{A}}^{conc}(R) \trianglelefteq \bot_{\mathcal{A}}^{abs}(R)$ By rule $\trianglelefteq - \texttt{bot}$

$\bot_{\mathcal{A}}^{conc} \trianglelefteq \bot_{\mathcal{A}}^{abs}$ By rule $\trianglelefteq - \rho$

□

Theorem F.3. Soundness of Constraint with Partial Substitution

$$
\begin{array}{ll}
\mathit{forall\ deriv.} & \\
& \mathcal{A}^{\mathrm{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathrm{abs}} \\
& \rho^{\mathrm{conc}} \sqsubseteq \rho^{\mathrm{abs}} \\
& \rho^{\mathrm{abs}}\ \mathsf{final} \\
& \rho^{\mathrm{conc}}\ \mathsf{final} \\
& \mathcal{A}^{\mathrm{abs}} \vdash \rho^{\mathrm{abs}}\ \mathsf{consistent} \\
& \mathcal{A}^{\mathrm{conc}} \vdash \rho^{\mathrm{conc}}\ \mathsf{consistent} \\
& \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \longrightarrow \rho^{\mathrm{abs}\Delta} \\
& \mathcal{B}^{\mathrm{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathrm{abs}} \\
\mathit{exists\ deriv.} & \\
& \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \longrightarrow \rho^{\mathrm{conc}\Delta} \\
& \rho^{\mathrm{conc}\Delta} \sqsubseteq \rho^{\mathrm{abs}\Delta} \\
& \rho^{\mathrm{conc}\Delta} \trianglelefteq \rho^{\mathrm{abs}\Delta}
\end{array}
$$

Proof:
By case analysis on $\mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{part}} \mathit{cons} \longrightarrow \rho^{\mathrm{abs}\Delta}$

Case:
$$
\dfrac{\begin{array}{c}
\mathit{cons} = \mathrm{op} : \mathrm{P}_{\mathrm{ctx}} \Rightarrow \mathrm{P}_{\mathrm{req}} \Downarrow \overline{\mathrm{Q}} \\
\Gamma_{\mathrm{y}} = \mathsf{FV}(\mathrm{op}) \cup \mathsf{FV}(\mathrm{P}_{\mathrm{ctx}}) \cup \mathsf{FV}(\overline{\mathrm{Q}}) \qquad \mathsf{allValidSubs}(\mathcal{A}^{\mathrm{abs}}; \sigma_{\mathrm{op}}; \Gamma_{\mathrm{y}}) = (\Sigma^{\mathrm{t}}_{\mathrm{a}}, \Sigma^{\mathrm{u}}_{\mathrm{a}}) \\
\Sigma^{\mathrm{t}}_{\mathrm{a}} \neq \varnothing \ \vee\ \Sigma^{\mathrm{u}}_{\mathrm{a}} \neq \varnothing \qquad \mathcal{P}^{\mathrm{t}}_{\mathrm{a}} = \{\rho^{\Delta} \mid \sigma \in \Sigma^{\mathrm{t}}_{\mathrm{a}} \ \wedge\ \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \\
\mathcal{P}^{\mathrm{u}}_{\mathrm{a}} = \{\updownarrow \rho^{\Delta} \mid \sigma \in \Sigma^{\mathrm{u}}_{\mathrm{a}} \ \wedge\ \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma \vdash_{\mathsf{full}} \mathit{cons} \hookrightarrow \rho^{\Delta}\} \qquad \mathcal{P}^{\Delta}_{\mathrm{a}} = \mathcal{P}^{\mathrm{t}}_{\mathrm{a}} \cup \mathcal{P}^{\mathrm{u}}_{\mathrm{a}}
\end{array}}{\mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}; \rho^{\mathrm{abs}}; \sigma_{\mathrm{op}} \vdash_{\mathsf{part}} \mathit{cons} \hookrightarrow (\ \sqcup\mkern-13mu=\ \mathcal{P}^{\Delta})}\ (\mathrm{BOUND})
$$

Let $\rho^{\Delta}_{\mathrm{a}} = \hookrightarrow (\ \sqcup\mkern-13mu=\ \mathcal{P}^{\Delta}_{\mathrm{a}})$	
$\mathsf{allValidSubs}(\mathcal{A}^{\mathrm{conc}}; \sigma_{\mathrm{op}}; \Gamma_{\mathrm{y}}) = (\Sigma^{\mathrm{t}}_{\mathrm{c}}, \Sigma^{\mathrm{u}}_{\mathrm{c}})$	By Lemma All Valid Subs sound and complete
$\Sigma^{\mathrm{t}}_{\mathrm{c}} \subseteq \Sigma^{\mathrm{t}}_{\mathrm{a}} \cup \Sigma^{\mathrm{u}}_{\mathrm{a}}$	By Lemma All Valid Subs sound and complete
$\Sigma^{\mathrm{u}}_{\mathrm{c}} \subseteq \Sigma^{\mathrm{u}}_{\mathrm{a}}$	By Lemma All Valid Subs sound and complete
$\Sigma^{\mathrm{t}}_{\mathrm{c}} \supseteq \Sigma^{\mathrm{t}}_{\mathrm{a}}$	By Lemma All Valid Subs sound and complete
$\forall\ \sigma \in \Sigma^{\mathrm{t}}_{\mathrm{c}}\ .\ \mathcal{A}^{\mathrm{conc}} \vdash \sigma\ \mathsf{validFor}\ \Gamma_{\mathrm{y}}$	By Lemma All Valid Subs sound and complete
$\forall\ \sigma \in \Sigma^{\mathrm{u}}_{\mathrm{c}}\ .\ \mathcal{A}^{\mathrm{conc}} \vdash \sigma\ \mathsf{validFor}\ \Gamma_{\mathrm{y}}$	By Lemma All Valid Subs sound and complete
$\forall\ \sigma \in \Sigma^{\mathrm{t}}_{\mathrm{c}}\ .\ \mathcal{A}^{\mathrm{conc}} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\mathrm{P}_{\mathrm{ctx}})$	By $\mathsf{FV}(\mathrm{P}_{\mathrm{ctx}}) \subseteq \Gamma_{\mathrm{y}}$
$\forall\ \sigma \in \Sigma^{\mathrm{u}}_{\mathrm{c}}\ .\ \mathcal{A}^{\mathrm{conc}} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\mathrm{P}_{\mathrm{ctx}})$	By $\mathsf{FV}(\mathrm{P}_{\mathrm{ctx}}) \subseteq \Gamma_{\mathrm{y}}$
By case analysis on the property $\Sigma^{\mathrm{t}}_{\mathrm{c}} \cup \Sigma^{\mathrm{u}}_{\mathrm{c}} = \varnothing$	

Case: $\Sigma^{\mathrm{t}}_{\mathrm{c}} \cup \Sigma^{\mathrm{u}}_{\mathrm{c}} = \varnothing$

$\Sigma^{\mathrm{t}}_{\mathrm{c}} = \varnothing$	By inversion of $\Sigma^{\mathrm{t}}_{\mathrm{c}} \cup \Sigma^{\mathrm{u}}_{\mathrm{c}} = \varnothing$
$\Sigma^{\mathrm{u}}_{\mathrm{c}} = \varnothing$	By inversion of $\Sigma^{\mathrm{t}}_{\mathrm{c}} \cup \Sigma^{\mathrm{u}}_{\mathrm{c}} = \varnothing$
$\mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}; \rho^{\mathrm{conc}}; \mathit{cons} \vdash \mathit{instr} \hookrightarrow \bot^{\mathrm{conc}}_{\mathcal{A}}$	By rule $\mathit{cant} - \mathit{bind}$

$\mathcal{A}^{\mathtt{conc}} \vdash \bot_{\mathcal{A}}^{\mathtt{conc}}$ consistent — By definition of $\bot_{\mathcal{A}}$

$\mathcal{A}^{\mathtt{abs}} \vdash \rho_a^{\Delta}$ consistent — By Lemma forall binding consistent

$\mathtt{dom}(\bot_{\mathcal{A}}^{\mathtt{conc}}) \subseteq \mathtt{dom}(\rho_a^{\Delta})$ — By lemma consistency and $\sqsubseteq_{\mathcal{A}}$ implies ρ domains subset

$\forall\, R \mapsto E_c \in \bot_{\mathcal{A}}^{\mathtt{conc}}$.

> $E_c = \mathtt{bot}$ — By definition of $\bot_{\mathcal{A}}$
>
> $E_c \sqsubseteq \rho_a^{\Delta}(R)$ — By rule $\sqsubseteq -\mathtt{bot}$

$\forall\, R \mapsto E_c \in \bot_{\mathcal{A}}^{\mathtt{conc}} .\ E_c \sqsubseteq \rho_a^{\Delta}(R)$ — By quantification above

$\bot_{\mathcal{A}}^{\mathtt{conc}} \sqsubseteq \rho_a^{\Delta}(R)$ — By rule $\sqsubseteq -\rho$

$\Sigma_a^t = \varnothing$ — By $\Sigma_c^t \supseteq \Sigma_a^t$ and $\Sigma_c^t = \varnothing$

$\mathcal{P}_a^t = \varnothing$ — By $|\mathcal{P}_a^t| = |\Sigma_a^t|$

$\forall\, \rho_a^{u\Delta} \in \mathcal{P}_a^u$.

> Let $\rho_a^{u\Delta} =\!\updownarrow \rho_a^{u\Delta'}$
>
> where $\mathcal{A}^{\mathtt{abs}}; \mathcal{B}^{\mathtt{abs}}; \rho^{\mathtt{abs}}; \sigma^u \vdash_{\mathsf{full}} \mathtt{cons} \hookrightarrow \rho_a^{u\Delta'}$ and $\sigma^u \in \Sigma_a^u$ — By construction of $\mathcal{P}_a^u$
>
> $\forall\, R \mapsto E \in \rho_a^{u\Delta} .\ E = \mathtt{bot} \vee E = \mathsf{unknown}$ — By $\updownarrow$ creates polarity

$\forall\, \rho_a^{u\Delta} \in \mathcal{P}_a^u .\ \forall\, R \mapsto E \in \rho_a^{u\Delta} .\ E = \mathtt{bot} \vee E = \mathsf{unknown}$ — By quantification

$\forall\, R \mapsto E \in \rho_a^{\Delta} .\ E = \mathtt{bot} \vee E = \mathsf{unknown}$ — By $\sqcup$ preserves polarity

$\forall\, R \in \mathtt{dom}(\bot_{\mathcal{A}^{\mathtt{conc}}})$.

> $\bot_{\mathcal{A}^{\mathtt{conc}}}(R) = \mathtt{bot}$ — By definition of $\bot$
>
> Let $E_a = \rho_a^{\Delta}(R)$
>
> Case analysis on the value of E_a
>
> > $E_a = \mathtt{bot}$
> >
> > $\mathtt{bot} \trianglelefteq \mathtt{bot}$ — By rule $\trianglelefteq - \mathtt{bot}$
> >
> > $E_a = \mathsf{unknown}$
> >
> > $\mathtt{bot} \trianglelefteq \mathsf{unknown}$ — By rule $\trianglelefteq - \mathsf{unknown}$
> >
> > $E_a = \mathtt{true}$
> >
> > Contradiction with $\forall\, R \mapsto E \in \rho_a^{\Delta} .\ E = \mathtt{bot} \vee E = \mathsf{unknown}$
> >
> > $E_a = \mathtt{false}$
> >
> > Contradiction with $\forall\, R \mapsto E \in \rho_a^{\Delta} .\ E = \mathtt{bot} \vee E = \mathsf{unknown}$

$\forall\, R \in \mathtt{dom}(\bot_{\mathcal{A}^{\mathtt{conc}}}) .\ \bot_{\mathcal{A}^{\mathtt{conc}}}(R) \trianglelefteq \rho_a^{\Delta}(R)$ — By quantification

$\bot_{\mathcal{A}^{\mathtt{conc}}} \trianglelefteq \rho_a^{\Delta}$ — By rule $\trianglelefteq - \rho$

Case: $\Sigma^t_c \cup \Sigma^u_c \neq \varnothing$

$\Sigma^t_c \neq \varnothing \lor \Sigma^u_c \neq \varnothing$ By inversion on $\cup$

Let $\mathcal{P}^t_c = \{\rho^\Delta \mid \sigma \in \Sigma^t_c \ \land\ \mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow \rho^\Delta\} \ \forall\ \sigma^t \in \Sigma^t_c$.

$\sigma^t \in \Sigma^t_a \cup \Sigma^u_a$ By inversion on $\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$

Case analysis on the location of σ^t

$\sigma^t \in \Sigma^t_a$

$\exists$ distinct $\rho^{t\Delta}_a \in \mathcal{P}^t_a \ .\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^t \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{t\Delta}_a$By the construction of $\mathcal{P}^t_a$ and $|\mathcal{P}^t_a| = |\Sigma$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^t \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{t\Delta}_c$ By lemma full constraint binding sound

$\mathcal{A}_{conc} \vdash \rho^{t\Delta}_c$ consistent By lemma full constraint consistent

$\rho^{t\Delta}_c \sqsubseteq \rho^{t\Delta}_a$ By lemma full constraint binding sound

$\rho^{t\Delta}_c \trianglelefteq \rho^{t\Delta}_a$ By lemma full constraint binding sound

$\rho^{t\Delta}_c \in \mathcal{P}^t_c$ By construction of $\mathcal{P}^t_c$

$\sigma^t \in \Sigma^u_a$

$\exists$ distinct $\rho^{u\Delta}_a \in \mathcal{P}^u_a \ .\ \rho^{u\Delta}_a =\updownarrow \rho^{u\Delta'}_a \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^t \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{u\Delta'}_a$By the construction of $\mathcal{P}$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^t \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{t\Delta}_c$ By lemma full constraint binding sound

$\mathcal{A}_{conc} \vdash \rho^{t\Delta}_c$ consistent By lemma full constraint consistent

$\rho^{t\Delta}_c \sqsubseteq \rho^{u\Delta'}_a$ By lemma full constraint binding sound

$\rho^{t\Delta}_c \trianglelefteq \rho^{u\Delta'}_a$ By lemma full constraint binding sound

$\rho^{t\Delta}_c \sqsubseteq \rho^{u\Delta}_a$ By lemma $\updownarrow$ on abs preserves $\sqsubseteq$

$\rho^{t\Delta}_c \trianglelefteq \rho^{u\Delta}_a$ By lemma $\updownarrow$ on abs preserves $\trianglelefteq$

$\rho^{t\Delta}_c \in \mathcal{P}^t_c$ By construction of $\mathcal{P}^u_c$

$\forall\ \sigma^t \in \Sigma^t_c \ .\ \exists$ distinct $\rho^t_c \in \mathcal{P}^t_c \ .\ \mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^t \vdash_{\mathsf{full}} cons \hookrightarrow \rho^t_c$ By quantification above

$|\mathcal{P}^t_c| = |\Sigma^t_c|$ By quantification above and construction of $\mathcal{P}^t_c$

$\forall\ \rho^{t\Delta}_c \in \mathcal{P}^t_c \ .\ \exists$ distinct $\rho^\Delta_a \in \mathcal{P}_a \ .\ \rho^{t\Delta}_c \sqsubseteq \rho^\Delta_a$ By quantification above

$\forall\ \rho^{t\Delta}_c \in \mathcal{P}^t_c \ .\ \exists$ distinct $\rho^\Delta_a \in \mathcal{P}_a \ .\ \rho^{t\Delta}_c \trianglelefteq \rho^\Delta_a$ By quantification above

$\forall\ \rho^{t\Delta}_c \in \mathcal{P}^t_c \ .\ \mathcal{A}_{conc} \vdash \rho^{t\Delta}_c$ consistent By quantification above

Let $\mathcal{P}^u_c = \{\updownarrow \rho^\Delta \mid \sigma \in \Sigma^u_c \ \land\ \mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow \rho^\Delta\} \ \forall\ \sigma^u \in \Sigma^u_c$.

$\sigma^u \in \Sigma^u_a$ By inversion on $\Sigma^u_c \subseteq \Sigma^u_a$

$\exists\ \rho^{u\Delta'}_a \in \mathcal{P}^u_a \ .\ \mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma^u \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{u\Delta'}_a$By the construction of $\mathcal{P}^u_a$ and $|\mathcal{P}^u_a| = |\Sigma^u_a|$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma^u \vdash_{\mathsf{full}} cons \hookrightarrow \rho^{u\Delta'}_c$ By lemma full constraint binding sound

$\mathcal{A}_{conc} \vdash \rho^{u\Delta'}_c$ consistent By lemma full constraint consistent

Let $\rho^{u\Delta}_a =\updownarrow \rho^{u\Delta'}_a$

$dom(\rho^{u\Delta'}_c) = dom(\rho^{conc})$By lemma consistency implies same domain Let $\rho^{u\Delta}_c =\updownarrow \rho^{u\Delta'}_c$

$\rho^{u\Delta'}_c \sqsubseteq \rho^{u\Delta'}_a$ By lemma full constraint binding sound

$\rho^{u\Delta'}_c \trianglelefteq \rho^{u\Delta'}_a$ By lemma full constraint binding sound

$\rho^{u\Delta}_c \sqsubseteq \rho^{u\Delta}_a$ By lemma $\updownarrow$ preserves $\sqsubseteq$

$\rho^{u\Delta}_c \trianglelefteq \rho^{u\Delta}_a$ By lemma $\updownarrow$ preserves $\trianglelefteq$

$\rho^{u\Delta}_c \in \mathcal{P}^u_c$ By construction of $\mathcal{P}^u_c$

$\forall\ \sigma^{u} \in \Sigma^{u}_{c}\ .\ \exists$ distinct $\rho^{u}_{c} \in \mathcal{P}^{u}_{c}\ .\ \mathcal{A}^{conc}; \mathcal{B}; \rho; \sigma^{u} \vdash_{\mathsf{part}} cons \hookrightarrow \rho^{u}_{c}$ By quantification above

$|\mathcal{P}^{u}_{c}| = |\Sigma^{u}_{c}|$ By quantification above and construction of $\mathcal{P}^{t}_{c}$

$\forall\ \rho^{u\Delta}_{c} \in \mathcal{P}^{u}_{c}\ .\ \exists$ distinct $\rho^{\Delta}_{a} \in \mathcal{P}_{a}\ .\ \rho^{u\Delta}_{c} \sqsubseteq \rho^{\Delta}_{a}$ By quantification above

$\forall\ \rho^{u\Delta}_{c} \in \mathcal{P}^{u}_{c}\ .\ \exists$ distinct $\rho^{\Delta}_{a} \in \mathcal{P}_{a}\ .\ \rho^{u\Delta}_{c} \trianglelefteq \rho^{\Delta}_{a}$ By quantification above

$\forall\ \rho^{u\Delta}_{c} \in \mathcal{P}^{u}_{c}\ .\ \mathcal{A}_{conc} \vdash \rho^{u\Delta}_{c}$ consistent By quantification above

Let $\mathcal{P}_{c} = \mathcal{P}^{t}_{c} \cup \mathcal{P}^{u}_{c}$

$\exists \bar{R}\ .\ \forall\ \rho_{c} \in \mathcal{P}_{c}\ .\ dom(\rho_{c}) = \bar{R}$ By inversion on consistency of each $\rho^{t\Delta}_{c}$

Let $\rho^{\Delta}_{c} = (\ \bigsqcup \mathcal{P}_{c})$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc} \vdash_{\mathsf{part}} cons \hookrightarrow \rho^{conc\Delta}$ By rule $bind$

$\forall\ \rho^{\Delta}_{c} \in \mathcal{P}_{c}\ .\ \exists$ distinct $\rho^{\Delta}_{a} \in \mathcal{P}_{a}\ .\ \rho^{\Delta}_{c} \sqsubseteq \rho^{\Delta}_{a}$ By $\mathcal{P}_{c} = \mathsf{Rho}^{t}_{c} \cup \mathsf{Rho}^{u}_{c}$

$\forall\ \rho^{\Delta}_{c} \in \mathcal{P}_{c}\ .\ \exists$ distinct $\rho^{\Delta}_{a} \in \mathcal{P}_{a}\ .\ \rho^{\Delta}_{c} \trianglelefteq \rho^{\Delta}_{a}$ By $\mathcal{P}_{c} = \mathsf{Rho}^{t}_{c} \cup \mathsf{Rho}^{u}_{c}$

$\rho^{\Delta}_{c} \sqsubseteq \rho^{\Delta}_{a}$ By $\bigsqcup$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

$\rho^{\Delta}_{c} \trianglelefteq \rho^{\Delta}_{a}$ By $\bigsqcup$ preserves $\sqsubseteq$ and $\trianglelefteq$ on sets

Case:
$$\frac{cons = op : P_{ctx} \Rightarrow P_{req} \Downarrow \overline{Q} \qquad \Gamma_{y} = \mathsf{FV}(op) \cup \mathsf{FV}(P_{ctx}) \cup \mathsf{FV}(\overline{Q}) \qquad \mathsf{allValidSubs}(\mathcal{A}^{abs}; \sigma_{op}; \Gamma_{y}) = (\varnothing, \varnothing)}{\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma_{op} \vdash_{\mathsf{part}} cons \hookrightarrow \bot_{\mathcal{A}^{abs}}}\ (\mathsf{CANT{-}BIND})$$

$\mathsf{allValidSubs}(\mathcal{A}^{conc}; \sigma_{op}; \Gamma_{y}) \mapsto (\Sigma^{t}_{c}, \Sigma^{u}_{c})$ By Lemma All Subs Sound

$\Sigma^{t}_{c} \subseteq \Sigma^{t}_{a} \cup \Sigma^{u}_{a}$ By Lemma All Subs Sound and complete

$\Sigma^{t}_{c} = \varnothing$ By inversion on $\subseteq$

$\Sigma^{u}_{c} \subseteq \Sigma^{u}_{a}$ By Lemma All Subs Sound and complete

$\Sigma^{u}_{c} = \varnothing$ By inversion on $\subseteq$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; cons \vdash instr \hookrightarrow \bot_{\mathcal{A}^{conc}}$ By rule $cant - bind$

$\mathcal{A}^{conc} \vdash \bot^{conc}_{\mathcal{A}}$ consistent By definition of $\bot_{\mathcal{A}}$

$\forall\ R \in dom(\bot^{conc}_{\mathcal{A}})\ .\ \bot^{conc}_{\mathcal{A}}(R) = \mathsf{bot}$ By definition of $\bot_{\mathcal{A}}$

$\mathcal{A}^{abs} \vdash \bot^{abs}_{\mathcal{A}}$ consistent By definition of $\bot_{\mathcal{A}}$

$\forall\ R \in dom(\bot^{abs}_{\mathcal{A}})\ .\ \bot^{abs}_{\mathcal{A}}(R) = \mathsf{bot}$ By definition of $\bot_{\mathcal{A}}$

$dom(\bot^{conc}_{\mathcal{A}}) \subseteq dom(\bot^{abs}_{\mathcal{A}}$ By Lemma consistency and $\sqsubseteq_{\mathcal{A}}$ implies ρ domains subset

$\forall\ R \in dom(\bot^{conc}_{\mathcal{A}})\ .\ \bot^{conc}_{\mathcal{A}}(R) \sqsubseteq \bot^{abs}_{\mathcal{A}}(R)$ By rule $\sqsubseteq -\mathsf{bot}$

$\bot^{conc}_{\mathcal{A}} \sqsubseteq \bot^{abs}_{\mathcal{A}}$ By rule $\sqsubseteq -\rho$

$\forall\ R \in dom(\bot^{conc}_{\mathcal{A}})\ .\ \bot^{conc}_{\mathcal{A}}(R) \trianglelefteq \bot^{abs}_{\mathcal{A}}(R)$ By rule $\trianglelefteq - \mathsf{bot}$

$\bot^{conc}_{\mathcal{A}} \trianglelefteq \bot^{abs}_{\mathcal{A}}$ By rule $\trianglelefteq - \rho$

□

Theorem F.4. Soundness of Constraint with Full Substitution

$$
\begin{array}{l}
\mathit{forall\ deriv.} \\
\qquad \mathcal{A}^{\mathsf{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathsf{abs}} \\
\qquad \mathcal{B}^{\mathsf{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathsf{abs}} \\
\qquad \rho^{\mathsf{conc}} \sqsubseteq \rho^{\mathsf{abs}} \\
\qquad \mathcal{A}^{\mathsf{abs}} \vdash \rho^{\mathsf{abs}}\ \mathsf{consistent} \\
\qquad \mathcal{A}^{\mathsf{conc}} \vdash \rho^{\mathsf{conc}}\ \mathsf{consistent} \\
\qquad \rho^{\mathsf{abs}}\ \mathsf{final} \\
\qquad \rho^{\mathsf{conc}}\ \mathsf{final} \\
\qquad \mathcal{A}^{\mathsf{conc}} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\mathsf{P}_{\mathsf{ctx}}) \\
\qquad \mathrm{dom}(\sigma) = \mathrm{dom}(\mathsf{FV}(\mathrm{cons})) \\
\qquad \mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \sigma \vdash_{\mathsf{full}} \mathrm{cons} \rightarrow \rho^{\mathsf{abs}\Delta} \\
\mathit{exists\ deriv.} \\
\qquad \mathcal{A}^{\mathsf{conc}}; \mathcal{B}^{\mathsf{conc}}; \rho^{\mathsf{conc}}; \sigma \vdash_{\mathsf{full}} \mathrm{cons} \rightarrow \rho^{\mathsf{conc}\Delta} \\
\qquad \rho^{\mathsf{conc}\Delta} \sqsubseteq \rho^{\mathsf{abs}\Delta} \\
\qquad \rho^{\mathsf{conc}\Delta} \trianglelefteq \rho^{\mathsf{abs}\Delta}
\end{array}
$$

Proof:
By case analysis on $\mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \sigma \vdash_{\mathsf{full}} \mathrm{cons} \hookrightarrow \rho^{\mathsf{abs}\Delta}$

Case: $$\frac{\mathrm{cons} = \mathrm{op} : \mathsf{P}_{\mathsf{ctx}} \Rrightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{False}}{\mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \sigma \vdash_{\mathsf{full}} \mathrm{cons} \hookrightarrow \bot_{\mathcal{A}^{\mathsf{abs}}}}\ (\mathrm{FULL{-}F{-}SOUND})$$

$$
\begin{array}{lr}
\mathcal{B}^{\mathsf{conc}}; \rho^{\mathsf{conc}} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathrm{t}^{\mathrm{c}} & \text{By lemma truth sound} \\
\mathrm{t}^{\mathrm{c}} \preccurlyeq \mathsf{False} & \text{By lemma truth sound} \\
\mathcal{B}^{\mathsf{conc}}; \rho^{\mathsf{conc}} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{False} & \text{By inversion on } \mathrm{t}^{\mathrm{c}} \preccurlyeq \mathsf{False} \\
\mathcal{A}^{\mathsf{conc}}; \mathcal{B}^{\mathsf{conc}}; \rho^{\mathsf{conc}}; \sigma \vdash_{\mathsf{full}} \mathrm{cons} \hookrightarrow \bot_{\mathcal{A}}^{\mathsf{conc}} & \text{By rule } \mathrm{full} - \mathrm{sound} - \mathsf{False} \\
\forall\ \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}}^{\mathsf{conc}}\ .\ \mathsf{E} = \mathbf{bot} & \text{By definition of } \bot \\
\forall\ \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}}^{\mathsf{conc}}\ .\ \mathsf{E} \sqsubseteq \bot_{\mathcal{A}}^{\mathsf{abs}}(\mathsf{R}) & \text{By rule } \sqsubseteq -\bot \\
\bot_{\mathcal{A}}^{\mathsf{conc}} \sqsubseteq \bot_{\mathcal{A}}^{\mathsf{abs}} & \text{By rule } \sqsubseteq \\
\forall\ \mathsf{R} \mapsto \mathsf{E} \in \bot_{\mathcal{A}}^{\mathsf{abs}}\ .\ \mathsf{E} = \mathbf{bot} & \text{By definition of } \bot \\
\forall\ \mathsf{R} \mapsto \mathsf{E} \in \rho^{\mathsf{conc}\Delta}\ .\ \mathsf{E} \trianglelefteq \rho^{\mathsf{abs}\Delta}(\mathsf{R}) & \text{By rule } \trianglelefteq - \bot \\
\bot_{\mathcal{A}}^{\mathsf{conc}} \trianglelefteq \bot_{\mathcal{A}}^{\mathsf{abs}} & \text{By rule } \trianglelefteq
\end{array}
$$

Case: $$\frac{\begin{array}{c}\mathrm{cons} = \mathrm{op} : \mathsf{P}_{\mathsf{ctx}} \Rrightarrow \mathsf{P}_{\mathsf{req}} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}} \vdash \mathsf{P}_{\mathsf{ctx}}[\sigma]\ \mathsf{True} \\ (\Sigma_{a}^{t}, \Sigma_{a}^{u}) = \mathsf{allValidSubs}(\mathcal{A}^{\mathsf{abs}}; \sigma; \mathsf{FV}(\mathrm{cons})) \\ \exists\ \sigma' \in \Sigma_{a}^{t}\ .\ \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}} \vdash \mathsf{P}_{\mathsf{req}}[\sigma']\ \mathsf{True}\end{array}}{\mathcal{A}^{\mathsf{abs}}; \mathcal{B}^{\mathsf{abs}}; \rho^{\mathsf{abs}}; \sigma \vdash_{\mathsf{full}} \mathrm{cons} \hookrightarrow \mathrm{lattice}(\bar{\mathsf{Q}}[\sigma])}\ (\mathrm{FULL{-}T{-}SOUND})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{P}_{ctx}[\sigma]\ t^{c}$	By lemma truth sound
$t^{c} \preccurlyeq \mathsf{True}$	By lemma truth sound
$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{P}_{ctx}[\sigma]\ \mathsf{True}$	By inversion on $t^{c} \preccurlyeq \mathsf{True}$
$(\Sigma^{t}_{c}, \Sigma^{u}_{c}) = \mathsf{allValidSubs}(\mathcal{A}^{conc}; \sigma; \mathsf{FV}(cons))$	By lemma valid subs Sound and Complete
$\forall\ \sigma \in \Sigma^{t}_{c} \cup \Sigma^{t}_{c}\ .\ \mathcal{A}^{conc} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(cons)$	By lemma valid subs Sound and Complete
$\forall\ \sigma \in \Sigma^{t}_{c} \cup \Sigma^{t}_{c}\ .\ \mathcal{A}^{conc} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\mathsf{P}_{req})$	By $\mathsf{FV}(\mathsf{P}_{req}) \subseteq \mathsf{FV}(cons)$
$\Sigma^{t}_{c} \subseteq \Sigma^{t}_{a} \cup \Sigma^{u}_{a}$	By lemma valid subs Sound and Complete
$\Sigma^{u}_{c} \subseteq \Sigma^{u}_{a}$	By lemma valid subs Sound and Complete
$\Sigma^{t}_{c} \supseteq \Sigma^{t}_{a}$	By lemma valid subs Sound and Complete
$\exists\ \sigma' \in \Sigma^{t}_{c}\ .\ \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{req}[\sigma']\ \mathsf{True}$	By $\Sigma^{t}_{c} \supseteq \Sigma^{t}_{a}$
Let $\rho^{abs\Delta} = lattice(\bar{\mathsf{Q}}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})$	
$\rho^{conc\Delta} = lattice(\bar{\mathsf{Q}}[\sigma], \mathcal{A}^{conc}, \mathcal{B}^{conc})$	By Lemma lattice sound
$\rho^{conc\Delta} \sqsubseteq \rho^{abs\Delta}$	By Lemma lattice sound
$\rho^{conc\Delta} \trianglelefteq \rho^{abs\Delta}$	By Lemma lattice sound
$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \rightarrow \rho^{conc\Delta}$	By rule $full - \mathsf{T} - sound$

Case:
$$\frac{\begin{array}{c} cons = \mathsf{op} : \mathsf{P}_{ctx} \Rightarrow \mathsf{P}_{req} \Downarrow \overline{\mathsf{Q}} \qquad \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{ctx}[\sigma]\ \mathsf{Unknown} \\ (\Sigma^{t}_{a}, \Sigma^{u}_{a}) = \mathsf{allValidSubs}(\mathcal{A}^{abs}; \sigma; \mathsf{FV}(cons)) \\ \exists\ \sigma' \in \Sigma^{t}_{a}\ .\ \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{req}[\sigma']\ \mathsf{True} \qquad \rho^{abs\Delta'} = lattice(\bar{\mathsf{Q}}[\sigma]) \end{array}}{\mathcal{A}^{abs}; \mathcal{B}^{abs}; \rho^{abs}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow\updownarrow \rho^{abs\Delta'}}\ (\text{FULL-U-SOUND})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{P}_{ctx}[\sigma]\ t$	By lemma truth sound
Case analysis on t	

Case: $t = \mathsf{True}$

$(\Sigma^{t}_{c}, \Sigma^{u}_{c}) = \mathsf{allValidSubs}(\mathcal{A}^{conc}; \sigma; \mathsf{FV}(cons))$	By lemma valid subs Sound and Complete
$\forall\ \sigma \in \Sigma^{t}_{c} \cup \Sigma^{u}_{c}\ .\ \mathcal{A}^{conc} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(cons)$	By lemma valid subs Sound and Complete
$\Sigma^{t}_{c} \subseteq \Sigma^{t}_{a} \cup \Sigma^{u}_{a}$	By lemma valid subs Sound and Complete
$\Sigma^{u}_{c} \subseteq \Sigma^{u}_{a}$	By lemma valid subs Sound and Complete
$\Sigma^{t}_{c} \supseteq \Sigma^{t}_{a}$	By lemma valid subs Sound and Complete
$\exists\ \sigma' \in \Sigma^{t}_{c}\ .\ \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{req}[\sigma']\ \mathsf{True}$	By $\Sigma^{t}_{c} \supseteq \Sigma^{t}_{a}$
Let $\rho^{abs\Delta'} = lattice(\bar{\mathsf{Q}}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})$	
$\rho^{conc\Delta} = lattice(\bar{\mathsf{Q}}[\sigma], \mathcal{A}^{conc}, \mathcal{B}^{conc})$	By Lemma lattice sound
$\rho^{conc\Delta} \sqsubseteq \rho^{abs\Delta'}$	By Lemma lattice sound
$\rho^{conc\Delta} \trianglelefteq \rho^{abs\Delta'}$	By Lemma lattice sound
Let $\rho^{abs\Delta} =\updownarrow \rho^{abs\Delta'}$	
$\rho^{conc\Delta} \sqsubseteq \rho^{abs\Delta}$	By Lemma $\updownarrow$ on abs preserves $\sqsubseteq$
$\rho^{conc\Delta} \trianglelefteq \rho^{abs\Delta}$	By Lemma $\updownarrow$ on abs preserves $\trianglelefteq$
$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \rightarrow \rho^{conc\Delta}$	By rule $full - \mathsf{T} - sound$

Case: $t = \mathsf{Unknown}$

$(\Sigma_c^t, \Sigma_c^u) = \mathsf{allValidSubs}(\mathcal{A}^{conc}; \sigma; \mathsf{FV}(cons))$ By lemma valid subs Sound and Complete

$\forall\ \sigma \in \Sigma_c^t \cup \Sigma_c^u\ .\ \mathcal{A}^{conc} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(cons)$ By lemma valid subs Sound and Complete

$\Sigma_c^t \subseteq \Sigma_a^t \cup \Sigma_a^u$ By lemma valid subs Sound and Complete

$\Sigma_c^u \subseteq \Sigma_a^u$ By lemma valid subs Sound and Complete

$\Sigma_c^t \supseteq \Sigma_a^t$ By lemma valid subs Sound and Complete

$\exists\ \sigma' \in \Sigma_c^t\ .\ \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}_{\mathsf{req}}[\sigma']\ \mathsf{True}$ By $\Sigma_c^t \supseteq \Sigma_a^t$

Let $\rho^{abs\Delta'} = lattice(\bar{R}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})$

$\rho^{conc\Delta'} = lattice(\bar{R}[\sigma], \mathcal{A}^{conc}, \mathcal{B}^{conc})$ By Lemma lattice sound

$\rho^{conc\Delta'} \sqsubseteq \rho^{abs\Delta'}$ By Lemma lattice sound

$\rho^{conc\Delta'} \trianglelefteq \rho^{abs\Delta'}$ By Lemma lattice sound

Let $\rho^{abs\Delta} =\updownarrow \rho^{abs\Delta'}$

Let $\rho^{conc\Delta} =\updownarrow \rho^{conc\Delta'}$

$\rho^{conc\Delta} \sqsubseteq \rho^{abs\Delta}$ By Lemma $\updownarrow$ preserves $\sqsubseteq$

$\rho^{conc\Delta} \trianglelefteq \rho^{abs\Delta}$ By Lemma $\updownarrow$ preserves $\trianglelefteq$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \rightarrow \rho^{conc\Delta}$ By rule $full - U - sound$

Case: $t = \mathsf{False}$

$\mathcal{A}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \hookrightarrow \bot_{\mathcal{A}}^{conc}$ By rule $full - sound - \mathsf{False}$

Let $\rho^{abs\Delta} =\updownarrow \rho^{abs\Delta'}$

$\forall\ R \mapsto E \in \bot_{\mathcal{A}}^{conc}\ .\ E = \mathtt{bot}$ By definition of $\bot$

$\forall\ R \mapsto E \in \bot_{\mathcal{A}}^{conc}\ .\ E \sqsubseteq \rho^{abs\Delta}(R)$ By rule $\sqsubseteq -\bot$

$\bot_{\mathcal{A}}^{conc} \sqsubseteq \rho^{abs\Delta}$ By rule $\sqsubseteq$

$\forall\ R \mapsto E \in \rho^{abs\Delta}\ .\ E = \mathtt{bot} \vee E = \mathtt{unknown}$ By $\updownarrow$ creates polarity

$\forall\ R \mapsto E \in \bot_{\mathcal{A}}^{conc}\ .\ E \trianglelefteq \rho^{abs\Delta}$ By rule $\trianglelefteq$

$\bot_{\mathcal{A}}^{conc} \trianglelefteq \rho^{abs\Delta}$ By rule $\trianglelefteq$

$\mathcal{A}^{conc}; \mathcal{B}^{conc}; \rho^{conc}; \sigma \vdash_{\mathsf{full}} cons \rightarrow \bot_{\mathcal{A}}^{conc}$ By rule $full - F - sound$

□

Theorem F.5. Truth Checking Sound

$$\begin{array}{ll}
\mathit{forall\ deriv.} & \\
& \rho^{conc} \sqsubseteq \rho^{abs} \\
& \mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs} \\
& \rho^{abs}\ \mathsf{final} \\
& \rho^{conc}\ \mathsf{final} \\
& \mathcal{A}^{conc} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\mathsf{P}) \\
& \mathcal{A}^{conc} \vdash \rho^{conc}\ \mathsf{consistent} \\
& \mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{P}[\sigma] t^{a} \\
\mathit{exists\ deriv.} & \\
& \mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{P}[\sigma] t^{c} \\
& t^{c} \preccurlyeq t^{a}
\end{array}$$

Proof:
By induction on $\rho^{abs} \vdash \mathsf{P}[\sigma]\ t^{a}$

Case: $\dfrac{\rho^{abs}(\mathsf{rel}(\bar{\ell})[\sigma]) = \texttt{true}}{\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{rel}(\bar{y})[\sigma]\ \mathsf{True}}\,(\mathsf{REL{-}TRUE})$

Let $\mathsf{R} = \mathsf{rel}(\bar{\ell})[\sigma]$
$\mathsf{R} \in dom(\rho^{conc})$ By lemma σ valid and ρ consistent
Let $\mathsf{E}^{c} = \rho^{conc}(\mathsf{R})$
By case analysis on the value of E^{c}

Case: $\mathsf{E}^{c} = \texttt{true}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{R}\ \mathsf{True}$ By rule $\mathsf{rel} - \mathsf{True}$
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $\mathsf{E}^{c} = \texttt{false}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $\mathsf{E}^{c} = \mathsf{unknown}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $\mathsf{E}^{c} = \texttt{bot}$

Contradiction with ρ^{conc} final

Case: $\dfrac{\rho^{abs}(\mathsf{rel}(\bar{\ell})[\sigma]) = \texttt{false}}{\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{rel}(\bar{y})[\sigma]\ \mathsf{True}}$ (REL−FALSE)

Let $R = \mathsf{rel}(\bar{\ell})[\sigma]$
$R \in dom(\rho^{conc})$ — By lemma σ valid and ρ consistent
Let $E^c = \rho^{conc}(R)$
By case analysis on the value of E^c

Case: $E^c = \texttt{false}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash R\ \mathsf{False}$ — By rule rel − False
$\mathsf{False} \preccurlyeq \mathsf{False}$ — By rule $\preccurlyeq - =$

Case: $E^c = \texttt{true}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $E^c = \mathtt{unknown}$

Contradiction with $\rho^{conc} \sqsubseteq \rho^{abs}$

Case: $E^c = \texttt{bot}$

Contradiction with ρ^{conc} final

Case: $\dfrac{\rho^{abs}(\mathsf{rel}(\bar{\ell})) = E^a \qquad E^a \neq \texttt{true} \qquad E^a \neq \texttt{false}}{\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{rel}(\bar{\ell})\ \mathsf{Unknown}}$ (REL−UNKNOWN−SOUND−COMPLETE)

$E_a = \mathtt{unknown}$ — By ρ^{abs} final
Let $R = \mathsf{rel}(\bar{\ell})[\sigma]$
$R \in dom(\rho^{conc})$ — By lemma σ valid and ρ consistent
Let $E^c = \rho^{conc}(R)$
By case analysis on the value of E^c

Case: $E^c = \texttt{false}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash R\ \mathsf{False}$ — By rule rel − False
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ — By rule $\preccurlyeq -U$

Case: $E^c = \texttt{true}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash R\ \mathsf{True}$ — By rule rel − False
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ — By rule $\preccurlyeq -U$

Case: $\mathsf{E}^c = \mathtt{unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{R}\ \mathsf{Unknown}$ By rule rel $-$ False
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathcal{U}$

Case: $\mathsf{E}^c = \mathtt{bot}$

Contradiction with ρ^{conc} final

Case: $$\dfrac{\mathcal{B}^{abs}; \rho \vdash \mathsf{A}\ t^a \qquad \mathcal{B}^{abs}(\ell_{test}) = t^a \qquad t^a \neq \mathsf{Unknown}}{\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{True}}\,(\mathsf{REL{-}TEST{-}TRUE})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{A}\ t^c$ By induction hypothesis
$t^c \preccurlyeq t^a$
By case analysis on t^c

Case: $t^c = \mathsf{True}$

$\mathcal{B}^{conc}(\ell_{test}) = \mathsf{True}$ By $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$
$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{True}$ By rule rel $-$ $test$ $-$ True
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $t^c = \mathsf{False}$

$\mathcal{B}^{conc}(\ell_{test}) = \mathsf{False}$ By $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$
$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{True}$ By rule rel $-$ $test$ $-$ True
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $t^c = \mathsf{Unknown}$

Contradiction with $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

Case: $$\dfrac{\mathcal{B}^{abs}; \rho \vdash \mathsf{A}\ t_1^a \qquad \mathcal{B}^{abs}(\ell_{test}) = t_2^a \qquad t_1^a \neq \mathsf{Unknown} t_2^a \neq \mathsf{Unknown} t_1^a \neq t_2^a}{\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{False}}\,(\mathsf{REL{-}TEST{-}FALSE})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{A}\ t_1^c$ By induction hypothesis
$t_1^c \preccurlyeq t_1^a$
By case analysis on t_1^c

Case: $t_1^c = \mathsf{True}$

$\mathcal{B}^{conc}(\ell_{test}) = t_2^c$ By $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$
By case analysis on t_2^c

Case: $t_2^c = \mathsf{True}$

Contradiction with $t_1^c \preccurlyeq t_1^a$ and $t_1^a \neq t_2^a$ and $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

Case: $t_2^c = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $\mathsf{rel} - test - \mathsf{False}$

$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Case: $t_2^c = \mathsf{Unknown}$

Contradiction with $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

Case: $t_1^c = \mathsf{False}$

$\mathcal{B}^{conc}(\ell_{test}) = t_2^c$ By $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

By case analysis on t_2^c

Case: $t_2^c = \mathsf{True}$

Contradiction with $t_1^c \preccurlyeq t_1^a$ and $t_1^a \neq t_2^a$ and $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

Case: $t_2^c = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $\mathsf{rel} - test - \mathsf{False}$

$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Case: $t_2^c = \mathsf{Unknown}$

Contradiction with $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

Case: $t_1^c = \mathsf{Unknown}$

Contradiction with $\mathcal{B}^{conc} \sqsubseteq \mathcal{B}^{abs}$

Case: $$\frac{\mathcal{B}^{abs}; \rho^{abs} \vdash A\ \mathsf{Unknown}}{\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}}\ (\mathsf{REL{-}TEST{-}U1})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A\ t^c$ By induction hypothesis

$t_1^c \preccurlyeq \mathsf{Unknown}$ By induction hypothesis

Let $t_2^c = \mathcal{B}^{conc}(\ell_{test})$ By case analysis on t_1^c

Case: $t_1^c = \mathsf{True}$

Let $t_2^c = \mathcal{B}^{conc}(\ell_{test})$

By case analysis on t_1^c

Case: $t^c_2 = \mathsf{True}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{True}$ By rule rel $-$ $test$ $-$ True
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t^c_2 = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}$ By rule rel $-$ $test$ $-$ False
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t^c_2 = \mathsf{Unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule rel $-$ $test$ $-$ $u2$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t^c_1 = \mathsf{False}$

Let $t^c_2 = \mathcal{B}^{conc}(\ell_{test})$
By case analysis on t^c_1

Case: $t^c_2 = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{True}$ By rule rel $-$ $test$ $-$ True
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t^c_2 = \mathsf{True}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}$ By rule rel $-$ $test$ $-$ False
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t^c_2 = \mathsf{Unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule rel $-$ $test$ $-$ $u2$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t^c_1 = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule rel $-$ $test$ $-$ $u1$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq - =$

Case: $$\frac{\mathcal{B}^{abs}(\ell_{test}) = \mathsf{Unknown} \qquad \mathcal{B}^{abs}; \rho^{abs} \vdash A\ t^a_1}{\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}}\,(\text{REL}-\text{TEST}-\text{U2})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A\ t_1^c$ By induction hypothesis
$t_1^c \preccurlyeq t_1^a$ By induction hypothesis
By case analysis on t_1^c

Case: $t_1^c = \mathsf{True}$

Let $t_2^c = \mathcal{B}^{conc}(\ell_{test})$
By case analysis on t_2^c

Case: $t_2^c = \mathsf{True}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{True}$ By rule $\mathsf{rel} - test - \mathsf{True}$
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t_2^c = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $\mathsf{rel} - test - \mathsf{False}$
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t_2^c = \mathsf{Unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $\mathsf{rel} - test - u2$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t_1^c = \mathsf{False}$

Let $t_2^c = \mathcal{B}^{conc}(\ell_{test})$
By case analysis on t_1^c

Case: $t_2^c = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{True}$ By rule $\mathsf{rel} - test - \mathsf{True}$
$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t_2^c = \mathsf{True}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{False}$ By rule $\mathsf{rel} - test - \mathsf{False}$
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t_2^c = \mathsf{Unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule $\mathsf{rel} - test - u2$
$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathsf{U}$

Case: $t_1^c = \mathsf{Unknown}$

$\mathcal{B}^{abs}; \rho^{abs} \vdash A/\ell_{test}\ \mathsf{Unknown}$ By rule rel $- \mathit{test} - u1$

$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq - =$

Case: $$\frac{\mathcal{B}^{abs}; \rho^{abs} \vdash \mathsf{S}\ \mathsf{Unknown}}{\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\ \mathsf{Unknown}}\ (\neg\mathsf{S}\text{—}\mathsf{UNKNOWN})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{S}\ t^c$ By induction hypothesis

$t^c \preccurlyeq \mathsf{Unknown}$ By induction hypothesis

By case analysis on the value of t^c

Case: $t^c = \mathsf{True}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{S}\ \mathsf{False}$ By rule $\neg\mathsf{S} - \mathsf{False}$

$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -U$

Case: $t^c = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{S}\ \mathsf{True}$ By rule $\neg\mathsf{S} - \mathsf{True}$

$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -U$

Case: $t^c = \mathsf{Unknown}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{S}\ \mathsf{Unknown}$ By rule $\neg\mathsf{S} - \mathsf{Unknown}$

$\mathsf{Unknown} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -U$

Case: $$\frac{\mathcal{B}^{abs}; \rho^{abs} \vdash S\mathsf{False}}{\mathcal{B}^{abs}; \rho^{abs} \vdash \neg\mathsf{S}\mathsf{True}}\ (\neg\mathsf{S}\text{—}\mathsf{TRUE})$$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \mathsf{S}\ t^c$ By induction hypothesis

$t^c \preccurlyeq \mathsf{False}$ By induction hypothesis

By case analysis on the value of t^c

Case: $t^c = \mathsf{False}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{S}\ \mathsf{True}$ By rule $\neg\mathsf{S} - \mathsf{True}$

$\mathsf{True} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -U$

Case: $t^c = \mathsf{True}$

Contradiction with $t^c \preccurlyeq \mathsf{False}$

Case: $\mathsf{t}^{c} = \mathsf{Unknown}$

Contradiction with $\mathsf{t}^{c} \preccurlyeq \mathsf{False}$

Case: $\dfrac{\mathcal{B}^{abs}; \rho^{abs} \vdash S\mathsf{True}}{\mathcal{B}^{abs}; \rho^{abs} \vdash \neg \mathsf{SFalse}}(\neg\text{R—FALSE})$

$\mathcal{B}^{conc}; \rho^{conc} \vdash S\ \mathsf{t}^{c}$ By induction hypothesis
$\mathsf{t}^{c} \preccurlyeq \mathsf{True}$ By induction hypothesis
By case analysis on the value of t^{c}

Case: $\mathsf{t}^{c} = \mathsf{True}$

$\mathcal{B}^{conc}; \rho^{conc} \vdash \neg\mathsf{S}\ \mathsf{False}$ By rule $\neg\mathsf{S} - \mathsf{False}$
$\mathsf{False} \preccurlyeq \mathsf{Unknown}$ By rule $\preccurlyeq -\mathcal{U}$

Case: $\mathsf{t}^{c} = \mathsf{False}$

Contradiction with $\mathsf{t}^{c} \preccurlyeq \mathsf{True}$

Case: $\mathsf{t}^{c} = \mathsf{Unknown}$

Contradiction with $\mathsf{t}^{c} \preccurlyeq \mathsf{True}$

Case: $\dfrac{}{\mathcal{B}^{abs}; \rho^{abs} \vdash true\mathsf{True}}(\text{TRUE})$

$\mathcal{B}^{conc}; \rho^{conc} \vdash true\mathsf{True}$ By rule $true$
$\mathsf{True} \preccurlyeq \mathsf{True}$ By rule $\preccurlyeq - =$

Case: $\dfrac{}{\mathcal{B}^{abs}; \rho^{abs} \vdash false\mathsf{False}}(\text{FALSE})$

$\mathcal{B}^{conc}; \rho^{conc} \vdash false\mathsf{False}$ By rule $false$
$\mathsf{False} \preccurlyeq \mathsf{False}$ By rule $\preccurlyeq - =$

Remaining cases work as expected for a three value logic.

□

Theorem F.6. Instruction Binding Sound

$$
\begin{aligned}
&\mathit{forall\ deriv.}\\
&\qquad \mathcal{A}^{\mathrm{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathrm{abs}}\\
&\qquad \mathcal{A}^{\mathrm{abs}} \vdash \mathit{instr} : \mathsf{op} \hookrightarrow (\Sigma^t_a, \Sigma^u_a)\\
&\mathit{exists\ deriv.}\\
&\qquad \mathcal{A}^{\mathrm{conc}} \vdash \mathit{instr} : \mathsf{op} \hookrightarrow (\Sigma^t_c, \Sigma^u_c)\\
&\qquad \Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a\\
&\qquad \Sigma^u_c \subseteq \Sigma^u_a\\
&\qquad \Sigma^t_c \supseteq \Sigma^t_a
\end{aligned}
$$

Proof:
By case analysis on the structure of the derivation of $\mathcal{A}^{\mathrm{abs}} \vdash \mathit{instr} : \mathsf{op} \hookrightarrow (\Sigma^t_a, \Sigma^u_a)$

Case: $$\frac{\begin{array}{c}\mathsf{FV}(\tau_{\mathsf{this}}.\mathsf{m}(\overline{\mathsf{y}} : \overline{\tau}) : \tau_{\mathsf{ret}}) \subseteq \Gamma_{\mathsf{y}}\\ (\Sigma^t_a, \Sigma^u_a) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{abs}}, \Gamma_{\mathsf{y}}, \{\mathtt{x}_{\mathtt{ret}}, \mathtt{x}_{\mathtt{this}}\} \cup \overline{\mathtt{x}}, \{\mathsf{ret}, \mathsf{this}\} \cup \overline{\mathsf{y}})\end{array}}{\mathcal{A}^{\mathrm{abs}}; \Gamma_{\mathsf{y}} \vdash \mathtt{x}_{\mathtt{ret}} = \mathtt{x}_{\mathtt{this}}.\mathsf{m}(\overline{\mathtt{x}}) : \tau_{\mathsf{this}}.\mathsf{m}(\overline{\mathsf{y}} : \overline{\tau}) : \tau_{\mathsf{ret}} \mapsto (\Sigma^t_a, \Sigma^u_a)}\,(\mathrm{INVOKE})$$

$(\Sigma^t_c, \Sigma^u_c) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{conc}}, \Gamma_{\mathsf{y}}, \{\mathtt{x}_{\mathtt{ret}}, \mathtt{x}_{\mathtt{this}}\} \cup \overline{\mathtt{x}}, \{\mathsf{ret}, \mathsf{this}\} \cup \overline{\mathsf{y}})$
By lemma FindLabels sound and complete

$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By lemma FindLabels sound and complete

$\Sigma^u_c \subseteq \Sigma^u_a$ By lemma FindLabels sound and complete

$\Sigma^t_c \supseteq \Sigma^t_a$ By lemma FindLabels sound and complete

$\mathcal{A}^{\mathrm{conc}} \vdash \mathtt{x}_{\mathtt{ret}} = \mathtt{x}_{\mathtt{this}}.\mathsf{m}(\overline{\mathtt{x}}) : \tau_{\mathsf{this}}.\mathsf{m}(\overline{\mathsf{y}} : \overline{\tau}) : \tau_{\mathsf{ret}} \hookrightarrow (\Sigma^t_c, \Sigma^u_c)$ By rule invoke

Case: $$\frac{\begin{array}{c}\mathsf{FV}(\ \mathsf{new}\ \ \tau(\overline{\mathsf{y}} : \overline{\tau})) \subseteq \Gamma_{\mathsf{y}}\\ (\Sigma^t_a, \Sigma^u_a) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{abs}}, \Gamma_{\mathsf{y}}, \{\mathtt{x}_{\mathtt{ret}}\} \cup \overline{\mathtt{x}}, \{\mathsf{this}\} \cup \overline{\mathsf{y}})\end{array}}{\mathcal{A}^{\mathrm{abs}}; \Gamma_{\mathsf{y}} \vdash \mathtt{x}_{\mathtt{ret}} =\ \mathtt{new}\ \ m(\overline{\mathtt{x}}) :\ \ \mathsf{new}\ \ \tau(\overline{\mathsf{y}} : \overline{\tau}) \mapsto (\Sigma^t_a, \Sigma^u_a)}\,(\mathrm{CONSTRUCTOR})$$

$(\Sigma^t_c, \Sigma^u_c) = \mathsf{findLabels}(\mathcal{A}^{\mathrm{conc}}, \Gamma_{\mathsf{y}}, \{\mathtt{x}_{\mathtt{ret}}, \mathtt{x}_{\mathtt{this}}\} \cup \overline{\mathtt{x}}, \{\mathsf{ret}, \mathsf{this}\} \cup \overline{\mathsf{y}})$
By lemma FindLabels sound and complete

$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By lemma FindLabels sound and complete

$\Sigma^u_c \subseteq \Sigma^u_a$ By lemma FindLabels sound and complete

$\Sigma^t_c \supseteq \Sigma^t_a$ By lemma FindLabels sound and complete

$\mathcal{A}^{\mathrm{conc}} \vdash \mathtt{x}_{\mathtt{ret}} =\ \mathtt{new}\ \ m(\overline{\mathtt{x}}) :\ \ \mathsf{new}\ \ \tau(\overline{\mathsf{y}} : \overline{\tau}) \hookrightarrow (\Sigma^t_c, \Sigma^u_c)$ By rule $\mathit{constructor}$

Case: $$\frac{}{\mathcal{A}^{\mathrm{abs}}; \Gamma_{\mathsf{y}} \vdash \mathit{eom} : \mathsf{end\text{-}of\text{-}method} \mapsto (\{\varnothing\}, \varnothing)}\,(\mathrm{EOM})$$

$\mathcal{A}^{\mathrm{conc}} \vdash \mathit{eom} : \mathsf{end\text{-}of\text{-}method} \mapsto (\{\varnothing\}, \varnothing)$ By rule eom

$\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By $\{\varnothing\} \subseteq \{\varnothing\} \cup \varnothing$

$\Sigma^u_c \subseteq \Sigma^u_a$ By $\varnothing \subseteq \varnothing$

$\Sigma^t_c \supseteq \Sigma^t_a$ By $\{\varnothing\} \supseteq \{\varnothing\}$

□

G Operator Lemmas

Theorem G.1. $\sqcup$ operator preserves $\sqsubseteq$

$$
\begin{array}{ll}
\mathit{foralld erivationsof} & \\
& \mathsf{E}_l^{\mathsf{conc}} \sqsubseteq \mathsf{E}_l^{\mathsf{abs}} \ \wedge\ \mathsf{E}_r^{\mathsf{conc}} \sqsubseteq \mathsf{E}_r^{\mathsf{abs}} \ \wedge \\
& \mathsf{E}_l^{\mathsf{conc}} \sqcup \mathsf{E}_r^{\mathsf{conc}} = \mathsf{E}^{\mathsf{conc}} \ \wedge\ \mathsf{E}_l^{\mathsf{abs}} \sqcup \mathsf{E}_r^{\mathsf{abs}} = \mathsf{E}^{\mathsf{abs}} \\
\mathit{existsderivationsof} & \\
& \mathsf{E}^{\mathsf{conc}} \sqsubseteq \mathsf{E}^{\mathsf{abs}}
\end{array}
$$

Proof:
By case analysis on structure of the derivation of $\mathsf{E}_l^{\mathsf{abs}} \sqcup \mathsf{E}_r^{\mathsf{abs}} = \mathsf{E}^{\mathsf{abs}}$

Case: $\dfrac{}{\mathtt{bot} \sqcup \mathsf{E} = \mathsf{E}}\,(\sqcup{-}\mathrm{BOT}{-}\mathrm{L})$

$\mathsf{E}_l^{\mathsf{conc}} = \mathtt{bot}$ By inversion on $\mathsf{E}_l^{\mathsf{conc}} \sqsubseteq \mathsf{E}_l^{\mathsf{abs}}$

$\mathsf{E}^{\mathsf{conc}} = \mathsf{E}_r^{\mathsf{conc}}$ By inversion on $\mathsf{E}_l^{\mathsf{conc}} \sqcup \mathsf{E}_r^{\mathsf{conc}} = \mathsf{E}^{\mathsf{conc}}$

$\mathsf{E}^{\mathsf{conc}} \sqsubseteq \mathsf{E}^{\mathsf{abs}}$ By equality

Case: $\dfrac{}{\mathsf{E} \sqcup \mathtt{bot} = \mathsf{E}}\,(\sqcup{-}\mathrm{BOT}{-}\mathrm{R})$

$\mathsf{E}_r^{\mathsf{conc}} = \mathtt{bot}$ By inversion on $\mathsf{E}_r^{\mathsf{conc}} \sqsubseteq \mathsf{E}_r^{\mathsf{abs}}$

$\mathsf{E}^{\mathsf{conc}} = \mathsf{E}_l^{\mathsf{conc}}$ By inversion on $\mathsf{E}_l^{\mathsf{conc}} \sqcup \mathsf{E}_r^{\mathsf{conc}} = \mathsf{E}^{\mathsf{conc}}$

$\mathsf{E}^{\mathsf{conc}} \sqsubseteq \mathsf{E}^{\mathsf{abs}}$ By equality

Case: $\dfrac{}{\mathsf{E} \sqcup \mathsf{E} = \mathsf{E}}\,(\sqcup{-}{=})$

$\mathsf{E}^{\mathsf{conc}} \sqsubseteq \mathsf{E}^{\mathsf{abs}}$ By equality

Case: $\dfrac{\mathsf{E}_l \neq \mathtt{bot} \qquad \mathsf{E}_r \neq \mathtt{bot} \qquad \mathsf{E}_l \neq \mathsf{E}_r}{\mathsf{E}_l \sqcup \mathsf{E}_r = \mathsf{unknown}}\,(\sqcup{-}{\neq})$

$\mathsf{E}^{\mathsf{conc}} \sqsubseteq \mathsf{E}^{\mathsf{abs}}$ By rule $\sqsubseteq$ $-$unknown

□

Theorem G.2. $\sqcup$ operator preserves $\trianglelefteq$

$$\begin{array}{l} \mathit{forall\ deriv.} \\ \qquad \mathsf{d1} : \mathsf{E}^{c}_{l} \trianglelefteq \mathsf{E}^{a}_{l} \\ \qquad \mathsf{d2} : \mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r} \\ \qquad \mathsf{d3} : \mathsf{E}^{a} = \mathsf{E}^{a}_{l} \sqcup \mathsf{E}^{a}_{r} \\ \qquad \mathsf{d4} : \mathsf{E}^{c} = \mathsf{E}^{c}_{l} \sqcup \mathsf{E}^{c}_{r} \\ \mathit{exists\ deriv.} \\ \qquad \mathsf{d5} : \mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a} \end{array}$$

Proof:
By case analysis on d4

Case: $\dfrac{}{\mathtt{bot} \sqcup \mathsf{E}^{c}_{r} = \mathsf{E}^{c}_{r}}\,(\sqcup\text{—BOT—L})$

By case analysis on d1

Case: $\dfrac{}{\mathtt{bot} \trianglelefteq \mathtt{bot}}\,(\trianglelefteq\text{—BOT})$

$\mathsf{E}^{a} = \mathsf{E}^{a}_{r}$	By inversion on d3
$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$	By $\mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r}$

Case: $\dfrac{}{\mathtt{bot} \trianglelefteq \mathtt{unknown}}\,(\trianglelefteq\text{—TOP})$

$\mathsf{E}^{a} = \mathtt{unknown}$	By inversion on d3
$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$	By rule $\trianglelefteq - \not\mathtt{bot}$ or $\trianglelefteq - \mathtt{unknown}$

Case: $\dfrac{\mathsf{E}^{c}_{l} \neq \mathtt{bot}}{\mathsf{E}^{c}_{l} \trianglelefteq \mathsf{E}^{a}_{l}}\,(\trianglelefteq\text{—OTHER})$

Invalid case by $\mathsf{E}^{c}_{l} = \mathtt{bot}$

Case: $\dfrac{}{\mathsf{E}^{c}_{l} \sqcup \mathtt{bot} = \mathsf{E}^{c}_{l}}\,(\sqcup\text{—BOT—R})$

By case analysis on d2

Case: $\dfrac{}{\mathtt{bot} \trianglelefteq \mathtt{bot}}\,(\trianglelefteq\text{—BOT})$

$\mathsf{E}^{a} = \mathsf{E}^{a}_{l}$	By inversion on d3
$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$	By $\mathsf{E}^{c}_{l} \trianglelefteq \mathsf{E}^{a}_{l}$

Case: $\dfrac{}{\mathtt{bot} \trianglelefteq \mathsf{unknown}}\,(\trianglelefteq\text{−TOP})$

$\mathsf{E}^{a} = \mathsf{unknown}$ By inversion on d3
$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By rule $\trianglelefteq - \not\mathtt{bot}$ or $\trianglelefteq - \mathsf{unknown}$

Case: $\dfrac{\mathsf{E}^{c}_{r} \neq \mathtt{bot}}{\mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r}}\,(\trianglelefteq\text{−OTHER})$

Invalid case by $\mathsf{E}^{c}_{r} = \mathtt{bot}$

Case: $\dfrac{}{\mathsf{E}^{c} \sqcup \mathsf{E}^{c} = \mathsf{E}^{c}}\,(\sqcup\text{−}=)$

By case analysis on d3

Case: $\dfrac{}{\mathtt{bot} \sqcup \mathsf{E}^{a}_{r} = \mathsf{E}^{a}_{r}}\,(\sqcup\text{−BOT−L})$

$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By $\mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r}$

Case: $\dfrac{}{\mathsf{E}^{a}_{l} \sqcup \mathtt{bot} = \mathsf{E}^{a}_{l}}\,(\sqcup\text{−BOT−R})$

$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By $\mathsf{E}^{c}_{l} \trianglelefteq \mathsf{E}^{a}_{l}$

Case: $\dfrac{}{\mathsf{E}^{a} \sqcup \mathsf{E}^{a} = \mathsf{E}^{a}}\,(\sqcup\text{−}=)$

$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By $\mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r}$

Case: $\dfrac{\mathsf{E}^{a}_{l} \neq \mathtt{bot} \qquad \mathsf{E}^{a}_{r} \neq \mathtt{bot} \qquad \mathsf{E}^{a}_{l} \neq \mathsf{E}^{a}_{r}}{\mathsf{E}^{a}_{l} \sqcup \mathsf{E}^{a}_{r} = \mathsf{unknown}}\,(\sqcup\text{−}\neq)$

By case on whether $\mathsf{E}^{c} = \mathtt{bot}$

Case: $\mathsf{E}^{c} = \mathtt{bot}$
$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By rule $\trianglelefteq - \mathsf{unknown}$

Case: $\mathsf{E}^{c} = \not\mathtt{bot}$
$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By rule $\trianglelefteq - \not\mathtt{bot}$

Case: $\dfrac{\mathsf{E}^{c}_{l} \neq \mathtt{bot} \qquad \mathsf{E}^{c}_{r} \neq \mathtt{bot} \qquad \mathsf{E}^{c}_{l} \neq \mathsf{E}^{c}_{r}}{\mathsf{E}^{c}_{l} \sqcup \mathsf{E}^{c}_{r} = \mathsf{unknown}}\,(\sqcup\text{−}\neq)$

$\mathsf{E}^{c} \trianglelefteq \mathsf{E}^{a}$ By rule $\trianglelefteq-\ \not{\mathtt{b}}\mathtt{ot}$

□

Theorem G.3. $\sqcup_{=}$ preserves polarity

$$\begin{array}{ll} \textit{forall deriv.} & \\ & d1 : E = E_l \sqcup_{=} E_r \\ & d2 : E_l = \mathtt{bot} \vee E_l = \mathsf{unknown} \\ \textit{exists deriv.} & \\ & d4 : E = \mathtt{bot} \vee E = \mathsf{unknown} \end{array}$$

Proof:

By case analysis on d1

Case: $\dfrac{}{E_l \sqcup_{=} E_l = E_l}(\text{EQJOIN}{-}{=})$

$E = \mathtt{bot} \vee E = \mathsf{unknown}$ By $E_l = \mathtt{bot} \vee E_l = \mathsf{unknown}$

Case: $\dfrac{E_l \neq E_r}{E_l \sqcup_{=} E_r = \mathsf{unknown}}(\text{EQJOIN}{-}{\neq})$

$E = \mathtt{bot} \vee E = \mathsf{unknown}$ By $E = \mathsf{unknown}$

□

Theorem G.4. $\boxed{=}$ less precise than operands

$$\begin{aligned}
&\mathit{forall\ deriv.}\\
&\qquad\qquad \mathrm{d1} : E = E_l \boxed{=} E_r\\
&\mathit{exists\ deriv.}\\
&\qquad\qquad \mathrm{d2} : E_l \sqsubseteq E\\
&\qquad\qquad \mathrm{d3} : E_r \sqsubseteq E
\end{aligned}$$

Proof:

By case analysis on d1

Case: $$\frac{}{E \boxed{=} E = E}\,(\text{EQJOIN}{-}{=})$$

$E_l \sqsubseteq E$By rule $\sqsubseteq - =$ $E_r \sqsubseteq E$ By rule $\sqsubseteq - =$

Case: $$\frac{E_l \neq E_r}{E_l \boxed{=} E_r = \text{unknown}}\,(\text{EQJOIN}{-}{\neq})$$

$E_l \sqsubseteq E$By rule $\sqsubseteq -$unknown $E_r \sqsubseteq E$ By rule $\sqsubseteq -$unknown

□

Theorem G.5. $\sqcup_{=}$ maintains super-precise on an operand

$$\begin{array}{ll} \textit{forall deriv.} & \\ & \mathrm{d1} : E = E_l \sqcup_{=} E_r \\ & \mathrm{d2} : E' \trianglelefteq E_l \\ \textit{exists deriv.} & \\ & \mathrm{d3} : E' \trianglelefteq E \end{array}$$

Proof:

By case analysis on d1

Case: $\dfrac{}{E_l \sqcup_{=} E_l = E_l}\,(\text{EQJOIN}{-}{=})$

$E' \trianglelefteq E$ By $E' \trianglelefteq E_l$

Case: $\dfrac{E_l \neq E_r}{E_l \sqcup_{=} E_r = \mathsf{unknown}}\,(\text{EQJOIN}{-}{\neq})$

$E' \trianglelefteq E$ By rule $\trianglelefteq - \mathsf{unknown}$ or $\trianglelefteq - \mathit{other}$

□

Theorem G.6. $\sqcup$ preserves $\sqsubseteq$ and $\trianglelefteq$

$$\begin{array}{l} \mathit{forall\ deriv.} \\ \qquad E^c = E^c_l \sqcup E^c_r \\ \qquad E^a = E^a_l \sqcup E^a_r \\ \qquad E^c_l \sqsubseteq E^a_l \\ \qquad E^c_r \sqsubseteq E^a_r \\ \qquad E^c_l \trianglelefteq E^a_l \\ \qquad E^c_r \trianglelefteq E^a_r \\ \mathit{exists\ deriv.} \\ \qquad E^c \sqsubseteq E^a \\ \qquad E^c \trianglelefteq E^a \end{array}$$

Proof:

By case analysis on $E^c = E^c_l \sqcup E^c_r$

Case: $\dfrac{}{E^c \sqcup E^c = E^c}\,(\text{EQJOIN}{-}{=})$

By case analysis on $E^a = E^a_l \sqcup E^a_r$

Case: $\dfrac{}{E^a \sqcup E^a = E^a}\,(\text{EQJOIN}{-}{=})$

$E^c \sqsubseteq E^a$ By $E^c_r \sqsubseteq E^a_r$
$E^c \trianglelefteq E^a$ By $E^c_r \trianglelefteq E^a_r$

Case: $\dfrac{E^a_l \neq E^a_r}{E^a_l \sqcup E^a_r = \mathsf{unknown}}\,(\text{EQJOIN}{-}{\neq})$

$E^c \sqsubseteq E^a$ By rule $\sqsubseteq -\mathsf{unknown}$
$E^c \trianglelefteq E^a$ By rule $\trianglelefteq - \mathsf{unknown}$ or $\trianglelefteq - \mathit{other}$

Case: $\dfrac{E^c_l \neq E^c_r}{E^c_l \sqcup E^c_r = \mathsf{unknown}}\,(\text{EQJOIN}{-}{\neq})$

$E^c \trianglelefteq E^a$ By rule $\trianglelefteq - \mathit{other}$
By case analysis on $E^a = E^a_l \sqcup E^a_r$

Case: $\dfrac{}{E^a \sqcup E^a = E^a}\,(\text{EQJOIN}{-}{=})$

By case analysis on the value of E^{a}

Case: $\mathsf{E}^{a} = \mathsf{unknown}$

$\mathsf{E}^{c} \sqsubseteq \mathsf{E}^{a}$ By rule $\sqsubseteq$ $-\mathsf{unknown}$

Case: $\mathsf{E}^{a} = \texttt{bot}$

$\mathsf{E}^{c}_{l} = \texttt{bot}$ By $\mathsf{E}^{c}_{l} \sqsubseteq \mathsf{E}^{a}_{l}$

$\mathsf{E}^{c}_{r} = \texttt{bot}$ By $\mathsf{E}^{c}_{r} \sqsubseteq \mathsf{E}^{a}_{r}$

Invalid case by $\mathsf{E}^{c}_{l} \neq \mathsf{E}^{c}_{r}$

Case: $\mathsf{E}^{a} = \texttt{true}$

$\mathsf{E}^{c}_{l} \neq \texttt{bot}$ By $\mathsf{E}^{c}_{l} \trianglelefteq \mathsf{E}^{a}_{l}$

$\mathsf{E}^{c}_{r} \neq \texttt{bot}$ By $\mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r}$

$\mathsf{E}^{c}_{l} = \texttt{true}$ By $\mathsf{E}^{c}_{l} \sqsubseteq \mathsf{E}^{a}_{l}$

$\mathsf{E}^{c}_{r} = \texttt{true}$ By $\mathsf{E}^{c}_{r} \sqsubseteq \mathsf{E}^{a}_{r}$

Invalid case by $\mathsf{E}^{c}_{l} \neq \mathsf{E}^{c}_{r}$

Case: $\mathsf{E}^{a} = \texttt{false}$

$\mathsf{E}^{c}_{l} \neq \texttt{bot}$ By $\mathsf{E}^{c}_{l} \trianglelefteq \mathsf{E}^{a}_{l}$

$\mathsf{E}^{c}_{r} \neq \texttt{bot}$ By $\mathsf{E}^{c}_{r} \trianglelefteq \mathsf{E}^{a}_{r}$

$\mathsf{E}^{c}_{l} = \texttt{false}$ By $\mathsf{E}^{c}_{l} \sqsubseteq \mathsf{E}^{a}_{l}$

$\mathsf{E}^{c}_{r} = \texttt{false}$ By $\mathsf{E}^{c}_{r} \sqsubseteq \mathsf{E}^{a}_{r}$

Invalid case by $\mathsf{E}^{c}_{l} \neq \mathsf{E}^{c}_{r}$

Case: $$\frac{\mathsf{E}^{a}_{l} \neq \mathsf{E}^{a}_{r}}{\mathsf{E}^{a}_{l} \sqcup\!\!\!\!= \mathsf{E}^{a}_{r} = \mathsf{unknown}}\,(\text{EQJOIN}-\neq)$$

$\mathsf{E}^{c} \sqsubseteq \mathsf{E}^{a}$ By rule $\sqsubseteq$ $-\mathsf{unknown}$

□

Theorem G.7. $\sqcup$ on sets preserves $\sqsubseteq$ and $\trianglelefteq$

forall der.

d1 : $\rho_c = \sqcup \mathcal{P}_c$

d2 : $\rho_a = \sqcup \mathcal{P}_a$

d3 : $\forall\, \rho'_c \in \mathcal{P}_c \,.\, \exists\, \rho'_a \in \mathcal{P}_a \,.\, \rho'_c \sqsubseteq \rho'_a \,\wedge\, \rho'_c \trianglelefteq \rho'_a$

(where each ρ'_c has a distinct ρ'_a)

exists der.

d4 : $\rho_c \sqsubseteq \rho_a$

d5 : $\rho_c \trianglelefteq \rho_a$

Proof:

By induction on d1

Case: $\rho_c = \rho'_c$

Let ρ'_a be the distinct ρ'_a for ρ'_c
By case analysis on the form of $\mathcal{P}_a$

Case $\mathcal{P}_a = \{\rho'_a\}$

$\rho_c \sqsubseteq \rho_a$ — By $\rho'_c \sqsubseteq \rho'_a$
$\rho_c \trianglelefteq \rho_a$ — By $\rho'_c \trianglelefteq \rho'_a$

Case $\mathcal{P}_a = \{\rho'_a\} \cup \mathcal{P}'_a$ where $\mathcal{P}'_a \neq \varnothing$

$\rho_a = \rho'_a \sqcup \rho''_a$ where $\rho''_a = \sqcup (\mathcal{P}_a - \rho'_a)$
$\rho'_a \sqsubseteq \rho_a$ — By Lemma $\sqcup$ less precise than operands
$\rho_c \sqsubseteq \rho_a$ — By $\sqsubseteq$ transitive
$\rho_c \trianglelefteq \rho_a$ — By Lemma $\sqcup$ maintains $\trianglelefteq$ for operand

Case: $\rho_c = \rho'_c \sqcup (\sqcup \mathcal{P}'_c)$

Let $\rho''_c = \sqcup \mathcal{P}'_c$
Let ρ'_a be the distinct ρ'_a for ρ'_c
$\rho_a = \rho'_a \sqcup \rho''_a$ where $\rho''_a = \sqcup (\mathcal{P}_a - \rho'_a)$
$\rho'_c \sqsubseteq \rho'_a$ — By induction hypothesis
$\rho'_c \trianglelefteq \rho'_a$ — By induction hypothesis
$\rho''_c \sqsubseteq \rho''_a$ — By induction hypothesis
$\rho''_c \trianglelefteq \rho''_a$ — By induction hypothesis
$\rho_c \sqsubseteq \rho_a$ — By Lemma $\sqcup$ preserves $\sqsubseteq$ and $\trianglelefteq$
$\rho_c \trianglelefteq \rho_a$ — By Lemma $\sqcup$ preserves $\sqsubseteq$ and $\trianglelefteq$

□

Theorem G.8. $\updownarrow$ creates polarity

$$\begin{array}{l} \mathit{forall\ der.} \\ \qquad \mathtt{d1} : \updownarrow \mathtt{E} = \mathtt{E}' \\ \mathit{exists\ der.} \\ \qquad \mathtt{d2} : \mathtt{E}' = \mathtt{bot} \vee \mathtt{E}' = \mathtt{unknown} \end{array}$$

Proof:

By case analysis on d1

Case: $\dfrac{}{\updownarrow \mathtt{bot} = \mathtt{bot}}\,(\updownarrow\text{–BOT})$

Case: $\dfrac{\mathtt{E} \neq \mathtt{bot}}{\updownarrow \mathtt{E} = \mathtt{unknown}}\,(\updownarrow\text{–UNKNOWN})$

□

Theorem G.9. $\updownarrow$ on abstract preserves $\sqsubseteq$

$$
\begin{aligned}
&\textit{forall der.}\\
&\qquad d1 : E^{c} \sqsubseteq E^{a'}\\
&\qquad d2 : \updownarrow E^{a'} = E^{a}\\
&\textit{exists der.}\\
&\qquad d2 : E^{c} \sqsubseteq E^{a}
\end{aligned}
$$

Proof:

By case analysis on d1

Case: $\dfrac{}{\mathsf{bot} \sqsubseteq E^{a'}}\,(\sqsubseteq\text{—BOT})$

$E^{c} \sqsubseteq E^{a}$ By rule $\sqsubseteq$ $-\mathsf{bot}$

Case: $\dfrac{}{E^{c} \sqsubseteq \mathsf{unknown}}\,(\sqsubseteq\text{—UNKNOWN})$

By case analysis on d2

Case: $\dfrac{}{\updownarrow \mathsf{bot} = \mathsf{bot}}\,(\updownarrow\text{—BOT})$

Invalid case since $E^{a'} = \mathsf{unknown}$

Case: $\dfrac{E^{a'} \neq \mathsf{bot}}{\updownarrow E^{a'} = \mathsf{unknown}}\,(\updownarrow\text{—UNKNOWN})$

$E^{c} \sqsubseteq E^{a}$ By rule $\sqsubseteq$ $-\mathsf{unknown}$

Case: $\dfrac{E^{a'} \neq \mathsf{bot} \qquad E^{a'} \neq \mathsf{unknown}}{E^{a'} \sqsubseteq E^{a'}}\,(\sqsubseteq\text{—=})$

By case analysis on d2

Case: $\dfrac{}{\updownarrow \mathsf{bot} = \mathsf{bot}}\,(\updownarrow\text{—BOT})$

$E^{c} \sqsubseteq E^{a}$ By rule $\sqsubseteq - =$

Case: $\dfrac{\mathsf{E}^{a'} \neq \mathtt{bot}}{\updownarrow \mathsf{E}^{a'} = \mathsf{unknown}}$ ($\updownarrow$–UNKNOWN)

$\mathsf{E}^{c} \sqsubseteq \mathsf{E}^{a}$ By rule $\sqsubseteq$ $-$unknown

□

Theorem G.10. $\updownarrow$ preserves $\sqsubseteq$

$$
\begin{array}{l}
\textit{forall der.} \\
\qquad d1 : \mathsf{E}_c' \sqsubseteq \mathsf{E}_a' \\
\qquad d2 : \mathsf{E}_a =\updownarrow \mathsf{E}_a' \\
\qquad d3 : \mathsf{E}_c =\updownarrow \mathsf{E}_c' \\
\textit{exists der.} \\
\qquad d4 : \mathsf{E}^c \sqsubseteq \mathsf{E}^a
\end{array}
$$

Proof:

By case analysis on d3

Case: $$\frac{}{\updownarrow \mathtt{bot} = \mathtt{bot}}\,(\updownarrow\text{—BOT})$$

$\mathsf{E}^c \sqsubseteq \mathsf{E}^a$	By rule $\sqsubseteq$ $-\mathtt{bot}$

Case: $$\frac{\mathsf{E}_c' \neq \mathtt{bot}}{\updownarrow \mathsf{E}_c' = \mathsf{unknown}}\,(\updownarrow\text{—UNKNOWN})$$

$\mathsf{E}_a' \neq \mathtt{bot}$	By inversion on $\mathsf{E}_c' \sqsubseteq \mathsf{E}_a'$
$\updownarrow \mathsf{E}_a' = \mathsf{unknown}$	By rule $\updownarrow$ $-\mathsf{unknown}$
$\mathsf{E}^c \sqsubseteq \mathsf{E}^a$	By rule $\sqsubseteq$ $-\mathsf{unknown}$

□

Theorem G.11. $\updownarrow$ creates $\trianglelefteq$

$$
\begin{array}{l}
\mathit{forall\ der.} \\
\qquad \mathrm{d1} : \mathsf{E}_a =\updownarrow \mathsf{E}'_a \\
\qquad \mathrm{d2} : \mathsf{E}_c =\updownarrow \mathsf{E}'_c \\
\mathit{exists\ der.} \\
\qquad \mathrm{d3} : \mathsf{E}^c \trianglelefteq \mathsf{E}^a
\end{array}
$$

Proof:

$\mathsf{E}^a = \text{unknown} \lor \mathsf{E}^a = \mathtt{bot}$ By lemma $\updownarrow$ creates polarity

$\mathsf{E}^c = \text{unknown} \lor \mathsf{E}^c = \mathtt{bot}$ By lemma $\updownarrow$ creates polarity

By case analysis on the value of E^c

Case: $\mathsf{E}^c = \mathtt{bot}$

By case analysis on the value of E^a

Case: $\mathsf{E}^a = \mathtt{bot}$

$\mathsf{E}^c \trianglelefteq \mathsf{E}^a$ By rule $\trianglelefteq - \mathtt{bot}$

Case: $\mathsf{E}^a = \text{unknown}$

$\mathsf{E}^c \trianglelefteq \mathsf{E}^a$ By rule $\trianglelefteq - \text{unknown}$

Case: $\mathsf{E}^c = \text{unknown}$

$\mathsf{E}^c \trianglelefteq \mathsf{E}^a$ By rule $\trianglelefteq - \mathsf{other}$

□

Theorem G.12. $\mathrel{\boxed{\leftarrow}}$ preserves $\sqsubseteq$

$$
\begin{array}{l}
\mathit{forall\ der.} \\
\qquad E_l^{conc} \sqsubseteq E_l^{abs} \;\wedge\; E_r^{conc} \sqsubseteq E_r^{abs} \;\wedge \\
\qquad E_l^{conc} \mathrel{\boxed{\leftarrow}} E_r^{conc} = E^{conc} \;\wedge\; E_l^{abs} \mathrel{\boxed{\leftarrow}} E_r^{abs} = E^{abs} \;\wedge \\
\qquad E_r^{conc} \trianglelefteq E_r^{abs} \\
\mathit{exists\ der.} \\
\qquad E^{conc} \sqsubseteq E^{abs}
\end{array}
$$

Proof:
Given $E_l^{conc} \sqsubseteq E_l^{abs}$, $E_r^{conc} \sqsubseteq E_r^{abs}$, $E_l^{conc} \mathrel{\boxed{\leftarrow}} E_r^{conc} = E^{conc}$, $E_l^{abs} \mathrel{\boxed{\leftarrow}} E_r^{abs} = E^{abs}$, $E_r^{conc} \trianglelefteq E_r^{abs}$
Show $E^{conc} \sqsubseteq E^{abs}$
By case analysis on the structure of the derivation of $E_l^{conc} \mathrel{\boxed{\leftarrow}} E_r^{conc} = E^{conc}$

Case: $\dfrac{}{E^{conc} \mathrel{\boxed{\leftarrow}} \mathtt{bot} = E^{conc}}\text{(OVRMEET}-\text{BOT)}$

By case analysis on the value of $E^{r}abs$

Case: $E_r^{abs} = \mathtt{bot}$

$E^{abs} = E_l^{abs}$ By inversion on $E_l^{abs} \mathrel{\boxed{\leftarrow}} E_r^{abs} = E^{abs}$
$E^{conc} \sqsubseteq E^{abs}$ By equality

Case: $E_r^{abs} = \mathsf{unknown}$

$E^{abs} = \mathsf{unknown}$ By inversion on $E_l^{abs} \mathrel{\boxed{\leftarrow}} E_r^{abs} = E^{abs}$
$E^{conc} \sqsubseteq E^{abs}$ By rule $\sqsubseteq -\mathsf{unknown}$

Case: $E_r^{abs} = \mathtt{true}$

Invalid case because $E_r^{conc} \trianglelefteq E_r^{abs}$

Case: $E_r^{abs} = \mathtt{false}$

Invalid case because $E_r^{conc} \trianglelefteq E_r^{abs}$

Case: $\dfrac{E_r^{conc} \neq \mathtt{bot}}{E_l^{conc} \mathrel{\boxed{\leftarrow}} E_r^{conc} = E_r^{conc}}\text{(OVRMEET}-\cancel{\text{B}}\text{OT)}$

$E_r^{abs} \neq \mathtt{bot}$ By inversion of $E_r^{conc} \sqsubseteq E_r^{abs}$
$E^{abs} = E_r^{abs}$ By inversion of $E_l^{abs} \mathrel{\boxed{\leftarrow}} E_r^{abs} = E^{abs}$
$E^{conc} \sqsubseteq E^{abs}$ By equality

□

Theorem G.13. Lattice with substitution is sound

$$\begin{aligned}
&\mathit{forall\ deriv.}\\
&\qquad d1 : \rho^{abs} = \mathsf{lattice}(\bar{Q}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs})\\
&\qquad d2 : \mathcal{A}^{conc} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{abs}\\
&\qquad d3 : \mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}\\
&\qquad d4 : \mathcal{A}^{conc} \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\bar{S})\\
&\mathit{exists\ deriv.}\\
&\qquad d5 : \rho^{conc} = \mathsf{lattice}(\bar{Q}[\sigma], \mathcal{A}^{conc}, \mathcal{B}^{conc})\\
&\qquad d6 : \rho^{conc} \sqsubseteq \rho^{abs}\\
&\qquad d7 : \rho^{conc} \trianglelefteq \rho^{abs}
\end{aligned}$$

Proof: By induction on d1:

Case: $$\frac{\rho^{a}_{1} = \mathsf{lattice}(Q[\sigma]; \mathcal{A}^{abs}; \mathcal{B}^{abs}) \qquad \rho^{a}_{2} = \mathsf{lattice}(\overline{Q}[\sigma]; \mathcal{A}^{abs}; \mathcal{B}^{abs})}{\mathsf{lattice}(Q[\sigma], \overline{Q}[\sigma]; \mathcal{A}^{abs}; \mathcal{B}^{abs}) = \rho^{a}_{1} \sqcup \rho^{a}_{2}}\ (\mathrm{LIST})$$

$\rho^{c}_{1} = \mathsf{lattice}(Q[\sigma]; \mathcal{A}^{conc}; \mathcal{B}^{conc})$	By induction hypothesis
$\rho^{c}_{1} \sqsubseteq \rho^{a}_{1}$	By induction hypothesis
$\rho^{c}_{1} \trianglelefteq \rho^{a}_{1}$	By induction hypothesis
$\rho^{c}_{2} = \mathsf{lattice}(\overline{Q}[\sigma]; \mathcal{A}^{conc}; \mathcal{B}^{conc})$	By induction hypothesis
$\rho^{c}_{2} \sqsubseteq \rho^{a}_{2}$	By induction hypothesis
$\rho^{c}_{2} \trianglelefteq \rho^{a}_{2}$	By induction hypothesis
Let $\rho^{c} = \rho^{c}_{1} \sqcup \rho^{c}_{2}$	
Let $\rho^{a} = \rho^{a}_{1} \sqcup \rho^{a}_{1}$	
$\rho^{c} \sqsubseteq \rho^{a}$	By Lemma $\sqcup$ preserves $\sqsubseteq$
$\rho^{c} \trianglelefteq \rho^{a}$	By Lemma $\sqcup$ preserves $\trianglelefteq$

Case: $$\frac{}{\mathsf{lattice}(A[\sigma]; \mathcal{A}^{abs}; \mathcal{B}^{abs}) = \bot_{\mathcal{A}^{abs}}[A[\sigma] \mapsto \mathtt{true}]}\ (\mathrm{LATTICE{-}R})$$

Let $R = A[\sigma]$	
Let $\rho^{a} = \bot_{\mathcal{A}^{abs}}[R \mapsto \mathtt{true}]$	
$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}^{conc}}\ \mathsf{consistent}$	By definition of $\bot_{\mathcal{A}}$
$R \in \mathsf{dom}(\bot_{\mathcal{A}^{conc}})$	By Lemma σ valid and ρ consistent gives ρ domain
Let $\rho^{c} = \bot_{\mathcal{A}^{conc}}[R \mapsto \mathtt{true}]$	
$\mathsf{lattice}(A[\sigma]; \mathcal{A}^{conc}; \mathcal{B}^{conc}) = \rho^{c}$	By rule $\mathsf{lattice} - R$
$\rho^{c} \sqsubseteq \rho^{a}$	By definition of $\bot_{\mathcal{A}}$
$\rho^{c} \trianglelefteq \rho^{a}$	By definition of $\bot_{\mathcal{A}}$

Case: $$\frac{}{\mathsf{lattice}(\neg A[\sigma]; \mathcal{A}^{abs}; \mathcal{B}^{abs}) = \bot_{\mathcal{A}^{abs}}[A[\sigma] \mapsto \mathtt{false}]}\ (\mathrm{LATTICE{-}\neg R})$$

Let $R = A[\sigma]$	
Let $\rho^a = \bot_{\mathcal{A}^{abs}}[R \mapsto \texttt{false}]$	
$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}^{conc}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$R \in \mathrm{dom}(\bot_{\mathcal{A}^{conc}})$	By Lemma σ valid and ρ consistent gives ρ domain
Let $\rho^c = \bot_{\mathcal{A}^{conc}}[R \mapsto \texttt{false}]$	
$\mathrm{lattice}(\neg A[\sigma]; \mathcal{A}^{conc}; \mathcal{B}^{conc}) = \rho^c$	By rule $\mathrm{lattice} - \neg R$
$\rho^c \sqsubseteq \rho^a$	By definition of $\bot_{\mathcal{A}}$
$\rho^c \trianglelefteq \rho^a$	By definition of $\bot_{\mathcal{A}}$

Case: $$\frac{\mathcal{B}^{abs}(\mathsf{y}_{\mathsf{test}}[\sigma]) = \mathsf{True}}{\mathrm{lattice}(A[\sigma]/\mathsf{y}_{\mathsf{test}}[\sigma], \mathcal{A}^{abs}, \mathcal{B}^{abs}) = \bot_{\mathcal{A}^{abs}}[A[\sigma] \mapsto \texttt{true}]}\ (\text{LATTICE--R--TEST--T})$$

Let $R = A[\sigma]$	
Let $\rho^a = \bot_{\mathcal{A}^{abs}}[R \mapsto \texttt{false}]$	
$\mathcal{A}^{conc} \vdash \bot_{\mathcal{A}^{conc}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$R \in \mathrm{dom}(\bot_{\mathcal{A}^{conc}})$	By Lemma σ valid and ρ consistent gives ρ domain
$\mathcal{B}^{conc}(\mathsf{y}_{\mathsf{test}}[\sigma]) = \mathsf{True}$	By $\mathcal{B}^{conc} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{abs}$
Let $\rho^c = \bot_{\mathcal{A}^{conc}}[R \mapsto \texttt{true}]$	
$\mathrm{lattice}(A[\sigma]/\mathsf{y}_{\mathsf{test}}[\sigma]; \mathcal{A}^{conc}; \mathcal{B}^{conc}) = \rho^c$	By rule $\mathrm{lattice} - R - \mathrm{test} - t$
$\rho^c \sqsubseteq \rho^a$	By definition of $\bot_{\mathcal{A}}$
$\rho^c \trianglelefteq \rho^a$	By definition of $\bot_{\mathcal{A}}$

Rest of the cases follow in a similar manner.

□

Theorem G.14. Lattice with substitution is complete

$$\begin{array}{ll}
\mathit{forall\ deriv.} & \\
& \rho^{\mathrm{conc}} = \mathtt{lattice}(\bar{Q}[\sigma], \mathcal{A}^{\mathrm{conc}}, \mathcal{B}^{\mathrm{conc}}) \\
& \mathcal{A}^{\mathrm{conc}} \sqsubseteq_{\mathcal{A}} \mathcal{A}^{\mathrm{abs}} \\
& \mathcal{B}^{\mathrm{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathrm{abs}} \\
\mathit{exists\ deriv.} & \\
& \rho^{\mathrm{abs}} = \mathtt{lattice}(\bar{Q}[\sigma], \mathcal{A}^{\mathrm{abs}}, \mathcal{B}^{\mathrm{abs}}) \\
& \rho^{\mathrm{conc}} \sqsubseteq \rho^{\mathrm{abs}} \\
& \rho^{\mathrm{conc}} \trianglelefteq \rho^{\mathrm{abs}}
\end{array}$$

Proof: By induction on d1:

Case: $$\frac{\rho_1^c = \mathtt{lattice}(Q[\sigma]; \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}) \qquad \rho_2^c = \mathtt{lattice}(\overline{Q}[\sigma]; \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}})}{\mathtt{lattice}(Q[\sigma], \overline{Q}[\sigma]; \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}) = \rho_1^c \sqcup \rho_2^c}\,(\mathrm{LIST})$$

$\rho_1^a = \mathtt{lattice}(Q[\sigma]; \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}})$	By induction hypothesis
$\rho_1^c \sqsubseteq \rho_1^a$	By induction hypothesis
$\rho_1^c \trianglelefteq \rho_1^a$	By induction hypothesis
$\rho_2^a = \mathtt{lattice}(\overline{Q}[\sigma]; \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}})$	By induction hypothesis
$\rho_2^c \sqsubseteq \rho_2^a$	By induction hypothesis
$\rho_2^c \trianglelefteq \rho_2^a$	By induction hypothesis
Let $\rho^c = \rho_1^c \sqcup \rho_2^c$	
Let $\rho^a = \rho_1^a \sqcup \rho_1^a$	
$\rho^c \sqsubseteq \rho^a$	By Lemma $\sqcup$ preserves $\sqsubseteq$
$\rho^c \trianglelefteq \rho^a$	By Lemma $\sqcup$ preserves $\trianglelefteq$

Case: $$\frac{}{\mathtt{lattice}(A[\sigma]; \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}) = \bot_{\mathcal{A}^{\mathrm{conc}}}[A[\sigma] \mapsto \mathtt{true}]}\,(\mathrm{LATTICE{-}R})$$

Let $R = A[\sigma]$	
$\mathtt{lattice}(R; \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}) = \bot_{\mathcal{A}^{\mathrm{abs}}}[R \mapsto \mathtt{true}]$	By rule $\mathtt{lattice} - R$
$\bot_{\mathcal{A}^{\mathrm{conc}}} \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}$	By definition of bot
$\bot_{\mathcal{A}^{\mathrm{conc}}} \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}$	By definition of bot
$\bot_{\mathcal{A}^{\mathrm{conc}}}[R \mapsto \mathtt{true}] \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}[R \mapsto \mathtt{true}]$	By rule $\sqsubseteq -\rho$
$\bot_{\mathcal{A}^{\mathrm{conc}}}[R \mapsto \mathtt{true}] \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}[R \mapsto \mathtt{true}]$	By rule $\trianglelefteq - \rho$

Case: $$\frac{}{\mathtt{lattice}(\neg A[\sigma]; \mathcal{A}^{\mathrm{conc}}; \mathcal{B}^{\mathrm{conc}}) = \bot_{\mathcal{A}^{\mathrm{conc}}}[A[\sigma] \mapsto \mathtt{false}]}\,(\mathrm{LATTICE{-}\neg R})$$

Let $R = A[\sigma]$	
$\mathtt{lattice}(\neg R; \mathcal{A}^{\mathrm{abs}}; \mathcal{B}^{\mathrm{abs}}) = \bot_{\mathcal{A}^{\mathrm{abs}}}[R \mapsto \mathtt{false}]$	By rule $\mathtt{lattice} - \neg R$

$\bot_{\mathcal{A}^{\mathrm{conc}}} \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}$ By definition of bot

$\bot_{\mathcal{A}^{\mathrm{conc}}} \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}$ By definition of bot

$\bot_{\mathcal{A}^{\mathrm{conc}}}[\mathsf{R} \mapsto \texttt{false}] \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \texttt{false}]$ By rule $\sqsubseteq -\rho$

$\bot_{\mathcal{A}^{\mathrm{conc}}}[\mathsf{R} \mapsto \texttt{false}] \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \texttt{false}]$ By rule $\trianglelefteq - \rho$

Case: $$\frac{\mathcal{B}^{\mathrm{conc}}(\mathsf{y}_{\mathsf{test}}[\sigma]) = \mathsf{True}}{lattice(A[\sigma]/\mathsf{y}_{\mathsf{test}}[\sigma], \mathcal{A}^{\mathrm{conc}}, \mathcal{B}^{\mathrm{conc}}) = \bot_{\mathcal{A}^{\mathrm{conc}}}[A[\sigma] \mapsto \texttt{true}]}\,(\text{LATTICE—R—TEST—T})$$

Let $\mathsf{R} = A[\sigma]$

Let $t_a = \mathcal{B}^{\mathrm{abs}}(\mathsf{y}_{\mathsf{test}}[\sigma])$

$\mathsf{True} \preccurlyeq t_a$ By $\mathcal{B}^{\mathrm{conc}} \sqsubseteq_{\mathcal{B}} \mathcal{B}^{\mathrm{abs}}$

By case analysis on t_a

Case: $t_a = \mathsf{True}$

$lattice(\mathsf{R}/\mathsf{y}_{\mathsf{test}}[\sigma], \mathcal{A}^{\mathrm{abs}}, \mathcal{B}^{\mathrm{abs}}) = \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \texttt{true}]$ By rule $lattice - R - test - t$

$\bot_{\mathcal{A}^{\mathrm{conc}}} \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}$ By definition of bot

$\bot_{\mathcal{A}^{\mathrm{conc}}} \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}$ By definition of bot

$\bot_{\mathcal{A}^{\mathrm{conc}}}[\mathsf{R} \mapsto \texttt{true}] \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \texttt{true}]$ By rule $\sqsubseteq -\rho$

$\bot_{\mathcal{A}^{\mathrm{conc}}}[\mathsf{R} \mapsto \texttt{true}] \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \texttt{true}]$ By rule $\trianglelefteq - \rho$

Case: $t_a = \mathsf{False}$

Invalid case by $\mathsf{True} \preccurlyeq t_a$

Case: $t_a = \mathsf{True}$

$lattice(\mathsf{R}/\mathsf{y}_{\mathsf{test}}[\sigma], \mathcal{A}^{\mathrm{abs}}, \mathcal{B}^{\mathrm{abs}}) = \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \mathsf{unknown}]$ By rule $lattice - R - test - u$

$\bot_{\mathcal{A}^{\mathrm{conc}}} \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}$ By definition of bot

$\bot_{\mathcal{A}^{\mathrm{conc}}} \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}$ By definition of bot

$\bot_{\mathcal{A}^{\mathrm{conc}}}[\mathsf{R} \mapsto \texttt{true}] \sqsubseteq \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \mathsf{unknown}]$ By rule $\sqsubseteq -\rho$

$\bot_{\mathcal{A}^{\mathrm{conc}}}[\mathsf{R} \mapsto \texttt{true}] \trianglelefteq \bot_{\mathcal{A}^{\mathrm{abs}}}[\mathsf{R} \mapsto \mathsf{unknown}]$ By rule $\trianglelefteq - \rho$

Rest of the cases follow in a similar manner.

□

Theorem G.15. σ valid and ρ consistent gives ρ domain

$$
\begin{array}{l}
\mathit{forall\ deriv.} \\
\qquad d1 :< \Gamma_\ell; \mathcal{L} > \vdash \sigma\ \mathsf{validFor}\ \mathsf{FV}(\mathsf{rel}(\bar{y})) \\
\qquad d2 :< \Gamma_\ell; \mathcal{L} > \vdash \rho\ \mathsf{consistent} \\
\mathit{exists\ deriv.} \\
\qquad d3 : \mathsf{rel}(\bar{y})[\sigma] \in \mathrm{dom}(\rho)
\end{array}
$$

Proof.

$\mathrm{dom}(\sigma) \supseteq \mathrm{dom}(\Gamma_y)$	By inversion on d1
$\forall\, y : \tau \in \Gamma_y \,.\, \exists\, \tau' \,.\, \tau' <: \Gamma_\ell(\sigma(y)) \ \wedge\ \tau' <: \tau$	By inversion on d1
$\mathrm{dom}(\rho) = \{\mathsf{rel}(\bar{\ell}) \mid \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ \lvert\bar{\tau}\rvert = \lvert\bar{\ell}\rvert = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma_\ell(\ell_i)\}$	By inversion on d2
$\bar{y} = \mathrm{dom}(\mathsf{FV}(\mathsf{rel}(\bar{y})))$	By inversion on FV
Let $\bar{\tau} = \mathcal{R}(\mathsf{rel})$	
$\bar{\ell} = \bar{y}[\sigma]$	By $\mathrm{dom}(\sigma) \supseteq \mathrm{dom}(\Gamma_y)$
$\lvert\bar{\ell}\rvert = \lvert\bar{y}\rvert = \lvert\bar{\tau}\rvert = n$	By substitution and typing of rel
Let $\Gamma_y = \mathsf{FV}(\mathsf{rel}(\bar{y}))$	
$\Gamma_y = y_0 : \tau_0, \ldots, y_n : \tau_n$	By inversion of FV
$\forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma_\ell(\ell_i)$	By $\mathrm{dom}(\sigma) \supseteq \mathrm{dom}(\Gamma_y)$
$\mathsf{rel}(\bar{y})[\sigma] \in \mathrm{dom}(\rho)$	By construction of the domain of ρ

□

Theorem G.16. $\sqcup$ preserves consistency

$$\begin{aligned}&\forall \mathcal{A}, \rho_l, \rho_r, \rho.\\ &\qquad \mathcal{A} \vdash \rho_l \textsf{ consistent } \wedge \mathcal{A} \vdash \rho_r \textsf{ consistent } \wedge \rho = \rho_l \sqcup \rho_r \implies\\ &\qquad \mathcal{A} \vdash \rho \textsf{ consistent}\end{aligned}$$

Proof:

Let $\mathcal{A} =< \Gamma_\ell; \mathcal{L} >$	
$\forall\, \textsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho_l)\ .\ \mathcal{R}(\textsf{rel}) = \bar{\tau} \wedge \lvert\bar{\tau}\rvert = \lvert\bar{\ell}\rvert \wedge \Gamma_\ell \textsf{ satisfies } \bar{\ell} : \bar{\tau}$	By inversion on $\mathcal{A} \vdash \rho_l$ consistent
$\mathrm{dom}(\rho_l) = \mathrm{dom}(\rho)$	By inversion on $\rho = \rho_l \sqcup \rho_r$
$\forall\, \textsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho)\ .\ \mathcal{R}(\textsf{rel}) = \bar{\tau} \wedge \lvert\bar{\tau}\rvert = \lvert\bar{\ell}\rvert \wedge \Gamma_\ell \textsf{ satisfies } \bar{\ell} : \bar{\tau}$	By $\mathrm{dom}(\rho_l) = \mathrm{dom}(\rho)$
$\mathcal{A} \vdash \rho \textsf{ consistent}$	By rule consistent

□

Theorem G.17. $\mathrel{\sqcup\mkern-13mu=}$ preserves consistency

$$\begin{aligned}&\forall \mathcal{A}, \rho_l, \rho_r, \rho.\\ &\qquad \mathcal{A} \vdash \rho_l \textsf{ consistent } \wedge \mathcal{A} \vdash \rho_r \textsf{ consistent } \wedge \rho = \rho_l \mathrel{\sqcup\mkern-13mu=} \rho_r \implies\\ &\qquad \mathcal{A} \vdash \rho \textsf{ consistent}\end{aligned}$$

Proof:

Let $\mathcal{A} =< \Gamma_\ell; \mathcal{L} >$	
$\forall\, \textsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho_l)\ .\ \mathcal{R}(\textsf{rel}) = \bar{\tau} \wedge \lvert\bar{\tau}\rvert = \lvert\bar{\ell}\rvert \wedge \Gamma_\ell \textsf{ satisfies } \bar{\ell} : \bar{\tau}$	By inversion on $\mathcal{A} \vdash \rho_l$ consistent
$\mathrm{dom}(\rho_l) = \mathrm{dom}(\rho)$	By inversion on $\rho = \rho_l \mathrel{\sqcup\mkern-13mu=} \rho_r$
$\forall\, \textsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho)\ .\ \mathcal{R}(\textsf{rel}) = \bar{\tau} \wedge \lvert\bar{\tau}\rvert = \lvert\bar{\ell}\rvert \wedge \Gamma_\ell \textsf{ satisfies } \bar{\ell} : \bar{\tau}$	By $\mathrm{dom}(\rho_l) = \mathrm{dom}(\rho)$
$\mathcal{A} \vdash \rho \textsf{ consistent}$	By rule consistent

□

Theorem G.18. $\mathrel{\sqcap\mkern-13mu\leftarrow}$ preserves consistency

$$\begin{aligned}&\forall \mathcal{A}, \mathcal{A}'\rho, \rho_\Delta, \rho'.\\ &\qquad \mathcal{A} \vdash \rho \textsf{ consistent } \wedge \mathcal{A}' \vdash \rho_\Delta \textsf{ consistent } \wedge \rho' = \rho \mathrel{\sqcup\mkern-13mu=} \rho_\Delta \implies\\ &\qquad \mathcal{A}' \vdash \rho' \textsf{ consistent}\end{aligned}$$

Proof:

Let $\mathcal{A}' =< \Gamma'_\ell ; \mathcal{L}' >$

$\forall\, \mathsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho_\Delta)\ .\ \mathcal{R}(\mathsf{rel}) = \bar{\tau}\ \wedge\ |\bar{\tau}| = |\bar{\ell}|\ \wedge\ \Gamma'_\ell\ \mathsf{satisfies}\ \bar{\ell} : \bar{\tau}$ By inversion on $\mathcal{A}' \vdash \rho_\Delta\ \mathsf{consistent}$

$\mathrm{dom}(\rho') = \mathrm{dom}(\rho_\Delta)$ By inversion on $\rho' = \rho \mathrel{\overline{|\!\leftarrow\!|}} \rho_\Delta$

$\forall\, \mathsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho)\ .\ \mathcal{R}(\mathsf{rel}) = \bar{\tau}\ \wedge\ |\bar{\tau}| = |\bar{\ell}|\ \wedge\ \Gamma'_\ell\ \mathsf{satisfies}\ \bar{\ell} : \bar{\tau}$ By $\mathrm{dom}(\rho_\iota) = \mathrm{dom}(\rho)$

$\mathcal{A}' \vdash \rho'\ \mathsf{consistent}$ By rule consistent

□

Theorem G.19. Transfer implies consistency

$$\begin{aligned}
&\forall\ deriv.\\
&\qquad \mathsf{d1} : \rho' = \mathsf{transfer}(\rho, \mathcal{A})\\
&\exists\ deriv.\\
&\qquad \mathsf{d2} : \mathcal{A} \vdash \rho'\ \mathsf{consistent}
\end{aligned}$$

Proof:

$\rho' = \{\mathsf{R} \mapsto \mathsf{E} \mid \mathsf{R} \in \mathrm{dom}(\bot_\mathcal{A})\ \wedge\ \mathsf{R} \in \mathrm{dom}(\rho) \implies \mathsf{E} = \rho(\mathsf{R})\ \wedge\ \mathsf{R} \notin \mathrm{dom}(\rho) \implies \mathsf{E} = \mathbf{unknown}\}$
By inversion on d1

$\mathrm{dom}(\rho') = \mathrm{dom}(\bot_\mathcal{A})$ By construction of ρ'

$\mathcal{A} \vdash \bot_\mathcal{A}\ \mathsf{consistent}$ By definition of $\bot_\mathcal{A}$

$\mathcal{A} \vdash \rho'\ \mathsf{consistent}$ By Lemma same domains imply same consistency

□

Theorem G.20. Lattice with substitution is consistent

$$
\begin{array}{ll}
\mathit{forall\ deriv.} & \\
& \rho = \mathit{lattice}(\bar{Q}[\sigma], \mathcal{A}, \mathcal{B}) \\
& \mathcal{A} \vdash \sigma \ \mathsf{validFor}\ \mathsf{FV}(\bar{Q}) \\
\mathit{exists\ deriv.} & \\
& \mathcal{A} \vdash \rho \ \mathsf{consistent}
\end{array}
$$

Proof: By induction on $\rho = \mathit{lattice}(\bar{Q}[\sigma], \mathcal{A}, \mathcal{B})$

Case: $\dfrac{\rho_1 = \mathit{lattice}(Q[\sigma]; \mathcal{A}; \mathcal{B}) \qquad \rho_2 = \mathit{lattice}(\overline{Q}[\sigma]; \mathcal{A}; \mathcal{B})}{\mathit{lattice}(Q[\sigma], \overline{Q}[\sigma]; \mathcal{A}; \mathcal{B}) = \rho_1 \sqcup \rho_2}\,(\mathrm{LIST})$

$\mathcal{A} \vdash \rho_1$ consistent	By induction hypothesis
$\mathcal{A} \vdash \rho_2$ consistent	By induction hypothesis
$\mathcal{A} \vdash \rho_1 \sqcup \rho_2$ consistent	By Lemma $\sqcup$ preserves consistency

Case: $\dfrac{}{\mathit{lattice}(A[\sigma]; \mathcal{A}; \mathcal{B}) = \bot_{\mathcal{A}}[A[\sigma] \mapsto \mathtt{true}]}\,(\mathrm{LATTICE{-}R})$

Let $R = A[\sigma]$	
$\mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$R \in \mathit{dom}(\bot_{\mathcal{A}})$	By Lemma σ valid and ρ consistent gives ρ domain
$\mathit{dom}(\bot_{\mathcal{A}}) = \mathit{dom}(\bot_{\mathcal{A}}[R \mapsto \mathtt{true}])$	By $R \in \mathit{dom}(\bot_{\mathcal{A}})$
$\mathcal{A} \vdash \bot_{\mathcal{A}}[R \mapsto \mathtt{true}]$ consistent	By rule $\mathit{consistent}$

Case: $\dfrac{}{\mathit{lattice}(\neg A[\sigma]; \mathcal{A}; \mathcal{B}) = \bot_{\mathcal{A}}[A[\sigma] \mapsto \mathtt{false}]}\,(\mathrm{LATTICE{-}\neg R})$

Let $R = A[\sigma]$	
$\mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$R \in \mathit{dom}(\bot_{\mathcal{A}})$	By Lemma σ valid and ρ consistent gives ρ domain
$\mathit{dom}(\bot_{\mathcal{A}}) = \mathit{dom}(\bot_{\mathcal{A}}[R \mapsto \mathtt{false}])$	By $R \in \mathit{dom}(\bot_{\mathcal{A}})$
$\mathcal{A} \vdash \bot_{\mathcal{A}}[R \mapsto \mathtt{false}]$ consistent	By rule $\mathit{consistent}$

Case: $\dfrac{\mathcal{B}(y_{\mathsf{test}}[\sigma]) = \mathsf{True}}{\mathit{lattice}(A[\sigma]/y_{\mathsf{test}}[\sigma], \mathcal{A}, \mathcal{B}) = \bot_{\mathcal{A}}[R \mapsto \mathtt{true}]}\,(\mathrm{LATTICE{-}R{-}TEST{-}T})$

Let $R = A[\sigma]$	
$\mathcal{A} \vdash \bot_{\mathcal{A}}$ consistent	By definition of $\bot_{\mathcal{A}}$
$R \in \mathit{dom}(\bot_{\mathcal{A}})$	By Lemma σ valid and ρ consistent gives ρ domain
$\mathit{dom}(\bot_{\mathcal{A}}) = \mathit{dom}(\bot_{\mathcal{A}}[R \mapsto \mathtt{true}])$	By $R \in \mathit{dom}(\bot_{\mathcal{A}})$
$\mathcal{A} \vdash \bot_{\mathcal{A}}[R \mapsto \mathtt{true}]$ consistent	By rule $\mathit{consistent}$

Rest of the cases follow in a similar manner.

□

Theorem G.21. Consistency implies same domain

$$\begin{aligned}
&\forall\ deriv.\\
&\qquad \mathrm{d1} :< \Gamma_\ell; \mathcal{L} > \vdash \rho_1 \text{ consistent}\\
&\qquad \mathrm{d2} :< \Gamma_\ell; \mathcal{L} > \vdash \rho_2 \text{ consistent}\\
&\exists\ deriv.\\
&\qquad \mathrm{dom}(\rho_1) = \mathrm{dom}(rho_2)
\end{aligned}$$

Proof:

$\mathrm{dom}(\rho_1) = \{\mathsf{rel}(\bar{\ell}) \mid \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ |\bar{\tau}| = |\bar{\ell}| = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma_\ell(\ell_i)\}$
By inversion on d1

$\mathrm{dom}(\rho_2) = \{\mathsf{rel}(\bar{\ell}) \mid \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ |\bar{\tau}| = |\bar{\ell}| = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma_\ell(\ell_i)\}$
By inversion on d2

$\mathrm{dom}(\rho_1) = \mathrm{dom}(\rho_2)$ By construction above

□

Theorem G.22. Consistency and $\sqsubseteq_{\mathcal{A}}$ implies domains are subset

$$\begin{aligned}
&\forall\ deriv.\\
&\qquad \mathrm{d1} :< \Gamma^c_\ell; \mathcal{L}^c > \vdash \rho^c \text{ consistent}\\
&\qquad \mathrm{d2} :< \Gamma^a_\ell; \mathcal{L}^a > \vdash \rho^a \text{ consistent}\\
&\qquad \mathrm{d3} :< \Gamma^c_\ell; \mathcal{L}^c > \sqsubseteq_{\mathcal{A}} < \Gamma^a_\ell; \mathcal{L}^a >\\
&\exists\ deriv.\\
&\qquad \mathrm{dom}(\rho^c) \subseteq \mathrm{dom}(rho^a)
\end{aligned}$$

Proof:

$\mathrm{dom}(\rho^c) = \{\mathsf{rel}(\bar{\ell}) \mid \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ |\bar{\tau}| = |\bar{\ell}| = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma^c_\ell(\ell_i)\}$By inversion on d1

$\mathrm{dom}(\rho^a) = \{\mathsf{rel}(\bar{\ell}) \mid \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ |\bar{\tau}| = |\bar{\ell}| = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma^a_\ell(\ell_i)\}$By inversion on d2

$\forall \mathsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho^c) \,.\, \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ |\bar{\tau}| = |\bar{\ell}| = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma^c_\ell(\ell_i)$
By construction of $\mathrm{dom}(\rho^c)$

$\mathrm{dom}(\Gamma^a_\ell) = \mathrm{dom}(\Gamma^c_\ell)$ By inversion on d3

$\forall\, \ell : \tau \in \Gamma^c_\ell .\, \tau <: \Gamma^a_\ell(\ell)$ By inversion on d3

$\forall\ \mathsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho^c) \,.\, \bar{\tau} = \mathcal{R}(\mathsf{rel}) \ \wedge\ |\bar{\tau}| = |\bar{\ell}| = n \ \wedge\ \forall\, i \in 1 \ldots n \,.\, \exists\, \tau' \,.\, \tau' <: \tau_i \ \wedge\ \tau' <: \Gamma^a_\ell(\ell_i)$By <: transitive

$\forall\ \mathsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho^c) \,.\, \mathsf{rel}(\bar{\ell}) \in \mathrm{dom}(\rho^a)$By construction of $\mathrm{dom}(\rho^a)$ $\mathrm{dom}(\rho^c) \subseteq \mathrm{dom}(\rho^a)$ By $\subseteq$

□

Theorem G.23. Find Labels Sound and Complete

$\mathit{forall\ deriv.}$

$d1 :< \Gamma^c_\ell, \mathcal{L}^c > \sqsubseteq_{\mathcal{A}} < \Gamma^a_\ell, \mathcal{L}^a >$

$d2 : |\overline{x}| = |\overline{y}| = n$

$\mathit{exists\ deriv.}$

$d3 : \mathsf{findLabels}(< \Gamma^a_\ell, \mathcal{L}^a >, \Gamma_y, \overline{x}, \overline{y}) = (\Sigma^t_a, \Sigma^u_a)$

$d4 : \mathsf{findLabels}(< \Gamma^c_\ell, \mathcal{L}^c >, \Gamma_y, \overline{x}, \overline{y}) = (\Sigma^t_c, \Sigma^u_c)$

$d5 : \Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$

$d6 : \Sigma^u_c \subseteq \Sigma^u_a$

$d7 : \Sigma^t_c \supseteq \Sigma^t_a$

Proof:

Let $\Sigma^t_a = \{(y_1 \mapsto \ell_1), \ldots, (y_n \mapsto \ell_n) \mid$
$\forall\, i \in 1 \ldots n \,.\, \mathcal{L}^a(\mathtt{x}_i) = \{\ell_i\} \ \wedge\ \Gamma^a_\ell(\ell_i) <: \Gamma_y(y_i)\}$

Let $\Sigma^u_a = \{(y_1 \mapsto \ell_1), \ldots, (y_n \mapsto \ell_n) \mid$
$\forall\, i \in 1 \ldots n \,.\, \ell_i \in \mathcal{L}^a(\mathtt{x}_i) \ \wedge\ \exists\, \tau' \,.\, \tau' <: \Gamma^a_\ell(\ell_i) \ \wedge\ \tau' <: \Gamma_y(y_i)\} - \Sigma^t_a$

d3: $\mathsf{findLabels}(< \Gamma^a_\ell, \mathcal{L}^a >, \Gamma_y, \overline{x}, \overline{y}) = (\Sigma^t_a, \Sigma^u_a)$ By rule $\mathit{findLabels}$

Let $\Sigma^t_c = \{(y_1 \mapsto \ell_1), \ldots, (y_n \mapsto \ell_n) \mid$
$\forall\, i \in 1 \ldots n \,.\, \mathcal{L}^c(\mathtt{x}_i) = \{\ell_i\} \ \wedge\ \Gamma^c_\ell(\ell_i) <: \Gamma_y(y_i)\}$

Let $\Sigma^u_c = \{(y_1 \mapsto \ell_1), \ldots, (y_n \mapsto \ell_n) \mid$
$\forall\, i \in 1 \ldots n \,.\, \ell_i \in \mathcal{L}^c(\mathtt{x}_i) \ \wedge\ \exists\, \tau' \,.\, \tau' <: \Gamma^c_\ell(\ell_i) \ \wedge\ \tau' <: \Gamma_y(y_i)\} - \Sigma^t_c$

d4: $\mathsf{findLabels}(< \Gamma^c_\ell, \mathcal{L}^c >, \Gamma_y, \overline{x}, \overline{y}) = (\Sigma^t_c, \Sigma^u_c)$ By rule $\mathit{findLabels}$

$\mathrm{dom}(\mathcal{L}^c) = \mathrm{dom}(\mathcal{L}^a)$ By inversion on d1

$\mathrm{dom}(\Gamma^c_\ell) = \mathrm{dom}(\Gamma^a_\ell)$ By inversion on d1

$\forall\, \ell' : \tau' \in \Gamma^c_\ell .\ \tau' <: \Gamma^a_\ell(\ell')$ By inversion on d1

$\forall\, \mathtt{x}' \mapsto \overline{\ell}' \in \mathcal{L}^c .\ \overline{\ell}' \subseteq \mathcal{L}^a(\mathtt{x}') \ \wedge\ \overline{\ell}' \neq \varnothing$ By inversion on d1

$\forall\, \ell \in \mathrm{dom}(\Gamma^c_\ell) \,.\, \Gamma^c_\ell(\ell) <: \Gamma^a_\ell(\ell)$ By rewriting

$\forall\, \mathtt{x} \in \mathrm{dom}(\mathcal{L}^c) \,.\, \mathcal{L}^c(\mathtt{x}) \subseteq \mathcal{L}^a(\mathtt{x}) \ \wedge\ \mathcal{L}^c(\mathtt{x}) \neq \varnothing$ By rewriting

$\forall\, \sigma \in \Sigma^t_c \,.$

$\forall\, i\, (1 \ldots n) \,.$

$(y_i \mapsto \ell_i) \in \sigma$ By $|\sigma| = n$

$\{\ell_i\} = \mathcal{L}^c(\mathtt{x}_i) \ \wedge\ \Gamma^c_\ell(\ell_i) <: \Gamma_y(y_i)$ By construction of σ

$\ell_i \in \mathcal{L}^a(\mathtt{x}_i) \ \wedge\ \Gamma^c_\ell(\ell_i) <: \Gamma_y(y_i)$ By $\mathcal{L}^c(\mathtt{x}_i) \subseteq \mathcal{L}^a(\mathtt{x}_i)$

$\ell_i \in \mathcal{L}^a(\mathtt{x}_i) \ \wedge\ \exists\, \tau' \,.\, \tau' <: \Gamma^a_\ell(\ell_i) \ \wedge\ \tau' <: \Gamma_y(y_i)$ By $\tau' = \Gamma^c_\ell(\ell_i)$ and $\Gamma^c_\ell(\ell_i) <: \Gamma^a_\ell(\ell_i)$

$\sigma \in \Sigma^t_a \cup \Sigma^u_a$ By quantification above d5: $\Sigma^{t}_{c} \subseteq \Sigma^{t}_{a} \cup \Sigma^{u}_{a}$ By quantification above

$\forall\, \sigma \in \Sigma^{u}_{c}$.

$\forall\, i\ (1 \ldots n)$.

$(y_i \mapsto \ell_i) \in \sigma$ By $|\sigma| = n$
$\ell_i \in \mathcal{L}^{c}(\mathtt{x_i}) \ \wedge\ \exists\, \tau' . \ \tau' <: \Gamma^{c}_{\ell}(\ell_i) \ \wedge\ \tau' <: \Gamma_{y}(y_i)$ By construction of σ
$\ell_i \in \mathcal{L}^{a}(\mathtt{x_i}) \ \wedge\ \exists\, \tau' . \ \tau' <: \Gamma^{c}_{\ell}(\ell_i) \ \wedge\ \tau' <: \Gamma_{y}(y_i)$ By $\mathcal{L}^{c}(\mathtt{x_i}) \subseteq \mathcal{L}^{a}(\mathtt{x_i})$
$\ell_i \in \mathcal{L}^{a}(\mathtt{x_i}) \ \wedge\ \exists\, \tau' . \ \tau' <: \Gamma^{a}_{\ell}(\ell_i) \ \wedge\ \tau' <: \Gamma_{y}(y_i)$ By $\Gamma^{c}_{\ell}(\ell_i) <: \Gamma^{a}_{\ell}(\ell_i)$

$\sigma \in \Sigma^{u}_{a}$ By quantification above

d6: $\Sigma^{u}_{c} \subseteq \Sigma^{u}_{a}$ By quantification above

$\forall\, \sigma \in \Sigma^{t}_{a}$.

$\forall\, i\ (1 \ldots n)$.

$(y_i \mapsto \ell_i) \in \sigma$ By $|\sigma| = n$
$\{\ell_i\} = \mathcal{L}^{a}(\mathtt{x_i}) \ \wedge\ \Gamma^{a}_{\ell}(\ell_i) <: \Gamma_{y}(y_i)$ By construction of σ
$\{\ell_i\} = \mathcal{L}^{c}(\mathtt{x_i}) \ \wedge\ \Gamma^{a}_{\ell}(\ell_i) <: \Gamma_{y}(y_i)$ By $\mathcal{L}^{c}(\mathtt{x_i}) \subseteq \mathcal{L}^{a}(\mathtt{x_i})$ and $\mathcal{L}^{c}(\mathtt{x_i}) \neq \varnothing$
$\{\ell_i\} = \mathcal{L}^{c}(\mathtt{x_i}) \ \wedge\ \Gamma^{a}_{\ell}(\ell_i) <: \Gamma_{y}(y_i)$ By $\Gamma^{c}_{\ell}(\ell_i) <: \Gamma^{a}_{\ell}(\ell_i)$

$\sigma \in \Sigma^{t}_{c}$ By quantification above

d7: $\Sigma^{t}_{c} \supseteq \Sigma^{t}_{a}$ By quantification above

□

Theorem G.24. All Valid Substitutions Sound and Complete

$$
\begin{aligned}
&\mathit{forall\ deriv.}\\
&\qquad d1 : < \Gamma^c_\ell; \mathcal{L}^c > \sqsubseteq_{\mathcal{A}} < \Gamma^a_\ell; \mathcal{L}^a >\\
&\mathit{exists\ deriv.}\\
&\qquad d2 : \mathsf{allValidSubs}(< \Gamma^a_\ell; \mathcal{L}^a >; \sigma; \Gamma_y) = (\Sigma^t_a, \Sigma^u_a)\\
&\qquad d3 : \mathsf{allValidSubs}(< \Gamma^c_\ell; \mathcal{L}^c >; \sigma; \Gamma_y) = (\Sigma^t_c, \Sigma^u_c)\\
&\qquad d4 : \forall\ \sigma \in \Sigma^t_a \cup \Sigma^u_a\ .\ < \Gamma^a_\ell; \mathcal{L}^a > \vdash \sigma\ \mathsf{validFor}\ \Gamma_y\\
&\qquad d5 : \forall\ \sigma \in \Sigma^t_c \cup \Sigma^u_c\ .\ < \Gamma^c_\ell; \mathcal{L}^c > \vdash \sigma\ \mathsf{validFor}\ \Gamma_y\\
&\qquad d6 : \Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a\\
&\qquad d7 : \Sigma^u_c \subseteq \Sigma^u_a\\
&\qquad d8 : \Sigma^t_c \supseteq \Sigma^t_a
\end{aligned}
$$

Proof:

Let $\Sigma^t_a = \{\sigma' \mid \sigma' \supseteq \sigma\ \wedge\ \mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)\ \wedge$
$\forall\, y \mapsto \ell \in \sigma'\ .\ \Gamma^a_\ell(\ell) <: \Gamma_y(y)\}$

Let $\Sigma^u_a = \{\sigma' \mid \sigma' \supseteq \sigma\ \wedge\ \mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)\ \wedge$
$\forall\, y \mapsto \ell \in \sigma'\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^a_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)\} - \Sigma^t_a$

d2: $\mathsf{allValidSubs}(< \Gamma^a_\ell; \mathcal{L}^a >; \sigma; \Gamma_y) = (\Sigma^t_a, \Sigma^u_a)$ By rule $\mathit{validSubs}$

$\forall \sigma \in \Sigma^t_a\ .\ \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma_y)\ \wedge\ \forall\, y \mapsto \ell \in \sigma\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^a_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)$By construction of Σ^t_a and $\tau' = \Gamma^a_\ell(\ell)$

$\forall \sigma \in \Sigma^u_a\ .\ \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma_y)\ \wedge\ \forall\, y \mapsto \ell \in \sigma\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^a_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)$ By construction of Σ^u_a

$\forall \sigma \in \Sigma^t_a \cup \Sigma^u_a\ .\ \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma_y)\ \wedge\ \forall\, y \mapsto \ell \in \sigma\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^a_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)$By $\cup$ and above predicates

d4: $\forall \sigma \in \Sigma^t_a \cup \Sigma^u_a\ .\ < \Gamma^a_\ell; \mathcal{L}^a > \vdash \sigma\ \mathsf{validFor}\ \Gamma_y$ By rule $\sigma - \mathit{valid}$

Let $\Sigma^t_c = \{\sigma' \mid \sigma' \supseteq \sigma\ \wedge\ \mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)\ \wedge$
$\forall\, y \mapsto \ell \in \sigma'\ .\ \Gamma^c_\ell(\ell) <: \Gamma_y(y)\}$

Let $\Sigma^u_c = \{\sigma' \mid \sigma' \supseteq \sigma\ \wedge\ \mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)\ \wedge$
$\forall\, y \mapsto \ell \in \sigma'\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^c_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)\} - \Sigma^t_c$

d3: $\mathsf{allValidSubs}(< \Gamma^c_\ell; \mathcal{L}^c >; \sigma; \Gamma_y) = (\Sigma^t_c, \Sigma^u_c)$ By rule $\mathit{validSubs}$

$\forall \sigma \in \Sigma^t_c\ .\ \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma_y)\ \wedge\ \forall\, y \mapsto \ell \in \sigma\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^c_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)$By construction of Σ^t_c and $\tau' = \Gamma^c_\ell(\ell)$

$\forall \sigma \in \Sigma^u_c\ .\ \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma_y)\ \wedge\ \forall\, y \mapsto \ell \in \sigma\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^c_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)$ By construction of Σ^u_c

$\forall \sigma \in \Sigma^t_c \cup \Sigma^u_a\ .\ \mathrm{dom}(\sigma) = \mathrm{dom}(\Gamma_y)\ \wedge\ \forall\, y \mapsto \ell \in \sigma\ .\ \exists\, \tau'\ .\ \tau' <: \Gamma^a_\ell(\ell)\ \wedge\ \tau' <: \Gamma_y(y)$By $\cup$ and above predicates

d5: $\forall \sigma \in \Sigma^t_c \cup \Sigma^u_c\ .\ < \Gamma^c_\ell; \mathcal{L}^c > \vdash \sigma\ \mathsf{validFor}\ \Gamma_y$ By rule $\sigma - \mathit{valid}$

$\mathrm{dom}(\mathcal{L}^c) = \mathrm{dom}(\mathcal{L}^a)$ By inversion on d1

$\mathrm{dom}(\Gamma^c_\ell) = \mathrm{dom}(\Gamma^a_\ell)$ By inversion on d1

$\forall\, \ell' : \tau' \in \Gamma^c_\ell .\ \tau' <: \Gamma^a_\ell(\ell')$ By inversion on d1

$\forall\, \mathtt{x}' \mapsto \bar{\ell}' \in \mathcal{L}^c .\ \bar{\ell}' \subseteq \mathcal{L}^a(\mathtt{x}')\ \wedge\ \bar{\ell}' \neq \varnothing$ By inversion on d1

$\forall\, \ell \in \mathrm{dom}(\Gamma^c_\ell)\ .\ \Gamma^c_\ell(\ell) <: \Gamma^a_\ell(\ell)$ By rewriting

$\forall\, \mathtt{x} \in \mathrm{dom}(\mathcal{L}^c)\ .\ \mathcal{L}^c(\mathtt{x}) \subseteq \mathcal{L}^a(\mathtt{x})\ \wedge\ \mathcal{L}^c(\mathtt{x}) \neq \varnothing$ By rewriting

$\forall\, \sigma' \in \Sigma^t_c\ .$

$\sigma' \supseteq \sigma$ By construction of σ'

$\mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)$ By construction of σ'
$\forall\, (y \mapsto \ell) \in \sigma'\,.$

$\Gamma^c_\ell(\ell) <: \Gamma_y(y)$ By construction of σ'
$\exists \tau'\,.\, \tau' <: \Gamma^a_\ell(\ell) \;\wedge\; \tau' <: \Gamma_y(y)$ By $\tau' = \Gamma^c_\ell(\ell)$ and $\Gamma^c_\ell(\ell_i) <: \Gamma^a_\ell(\ell_i)$

$\forall\, (y \mapsto \ell) \in \sigma'\,.\, \exists \tau'\,.\, \tau' <: \Gamma^a_\ell(\ell) \;\wedge\; \tau' <: \Gamma_y(y)$
$\sigma' \in \Sigma^t_a \cup \Sigma^u_a$ By construction of Σ^t_a and Σ^u_a

d4: $\Sigma^t_c \subseteq \Sigma^t_a \cup \Sigma^u_a$ By quantification above

$\forall\, \sigma' \in \Sigma^u_c\,.$

$\sigma' \supseteq \sigma$ By construction of σ'
$\mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)$ By construction of σ'
$\forall\, (y \mapsto \ell) \in \sigma'\,.$

$\exists\, \tau'\,.\, \tau' <: \Gamma^c_\ell(\ell) \;\wedge\; \tau' <: \Gamma_y(y)$ By construction of σ'
$\exists\, \tau'\,.\, \tau' <: \Gamma^a_\ell(\ell) \;\wedge\; \tau' <: \Gamma_y(y)$ By $\Gamma^c_\ell(\ell) <: \Gamma^a_\ell(\ell)$

$\forall\, (y \mapsto \ell) \in \sigma'\,.\, \exists \tau'\,.\, \tau' <: \Gamma^a_\ell(\ell) \;\wedge\; \tau' <: \Gamma_y(y)$
$\sigma' \in \Sigma^u_a$ By construction of Σ^u_a

d5: $\Sigma^u_c \subseteq \Sigma^u_a$ By quantification above

$\forall\, \sigma \in \Sigma^t_a\,.$

$\sigma' \supseteq \sigma$ By construction of σ'
$\mathrm{dom}(\sigma') = \mathrm{dom}(\Gamma_y)$ By construction of σ'
$\forall\, (y \mapsto \ell) \in \sigma'\,.$

$\Gamma^a_\ell(\ell) <: \Gamma_y(y)$ By construction of σ'
$\Gamma^c_\ell(\ell) <: \Gamma_y(y)$ By $\Gamma^c_\ell(\ell_i) <: \Gamma^a_\ell(\ell_i)$

$\forall\, (\mathsf{y} \mapsto \ell) \in \sigma' \;.\; \Gamma^{\mathsf{c}}_{\ell}(\ell) <: \Gamma_{\mathsf{y}}(\mathsf{y})$

$\sigma' \in \Sigma^{\mathsf{t}}_{\mathsf{c}}$ By construction of $\Sigma^{\mathsf{t}}_{\mathsf{c}}$

d6: $\Sigma^{\mathsf{t}}_{\mathsf{c}} \supseteq \Sigma^{\mathsf{t}}_{\mathsf{a}}$ By quantification above

□